Perbandingan Performa Progressive Web Apps dan Mobile Web Terkait Waktu Respon, Penggunaan Memori dan Penggunaan Media Penyimpanan

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Muhammad Rasyid Ridho
NIM: 125150207111036

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PERBANDINGAN PERFORMA PROGRESSIVE WEB APPS DAN MOBILE WEB TERKAIT WAKTU RESPON, PENGGUNAAN MEMORI DAN PENGGUNAAN MEDIA PENYIMPANAN

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Muhammad Rasyid Ridho
NIM: 125150207111036

Skripsi ini telah diuji dan dinyatakan lulus pada
15 Januari 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Aryo Pinandito, S.T, M.MT	Ratih Kartika Dewi, S.T., M.Kom
NIP: 19830519 201404 1 001	NIK: 201503 890520 2 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 15 Januari 2018

Muhammad Rasyid Ridho

NIM: 125150207111036

KATA PENGANTAR

Puji Syukur kehadirat Allah SWT, berkat rahmat dan karunia-Nya penulis dapat menyelesaikan penyusunan Skripsi dengan judul "Perbandingan Performa Progressive Web Apps dan Mobile Web Terkait Waktu Respon, Penggunaan Memori dan Penggunaan Media Penyimpanan". Skripsi ini diajukan sebagai salah satu syarat untuk mendapatkan gelar sarjana pada Program Studi Informatika Jurusan Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya Malang.

Penulis menyadari bahwa penyusunan skripsi ini tidak akan dapat dilakukan tanpa adanya bantuan dari beberapa pihak, oleh karena itu penulis menyampaikan rasa terima kasih dan penghargaan yang sebesar-besarnya kepada:

1. Bapak Aryo Pinandito, ST, M.MT selaku Dosen Pembimping I yang telah meberikan bimbingan, arahan dan motivasi hingga skripsi ini dapat terselesaikan.
2. Ibu Ratih Kartika Dewi, S.T., M.Kom selaku Dosen Pembimbing II yang telah memberikan bimbingan, pengetahuan dan motivasi untuk kesempurnaan penulisan skripsi ini.
3. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya, Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika dan Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Universitas Brawijaya.
4. Kedua orang tua yang selalu memberi dukungan moril maupun materil yang sangat membantu penulis dalam menempuh studi dan penyusunan skripsi ini.
5. Seluruh dosen Teknik Informatika yang telah memberikan ilmu serta wawasan dalam perkuliahan.
6. Ainin Nur Asiyah atas dukungan, motivasi, semangat dan doa yang diberikan kepada penulis sehingga terselesaikannya penulisan skripsi ini.
7. Seluruh teman-teman Informatika Universitas Brawijaya Angkatan 2012 atas dukungan, masukan dan semangat yang diberikan kepada penulis.
8. Teman-teman pengurus Himpunan Mahasiswa Informatika periode 2015/2016 yang telah memberikan semangat dan inspiriasi kepada penulis.
9. Teman-teman anggota Himpunan Mahasiswa Informatika serta Keluarga Besar Mahasiswa Fakultas Ilmu Komputer yang telah menemani penulis selama menjalani studi dan penyusunan skripsi ini.
10. Seluruh civitas akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan bantuan selama penulis menempuh studi di Fakultas Ilmu Komputer Universitas Brawijaya.
11. Semua pihak yang tidak dapat penulis sebutkan satu persatu, yang telah memberikan bantuan-bantuan yang sangat berharga selama menjalani studi dan menyelesaikan skripsi ini.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan, sehingga saran dan kritik yang membangun sangat penulis harapkan. Akhir kata penulis berharap skripsi ini dapat membawa manfaat bagi seluruh pihak yang menggunakannya.

Malang, 15 Januari 2018

Penulis,

Email: muh.r.ridho@gmail.com

ABSTRAK

Saat ini banyak teknologi-teknologi baru yang bermunculan di bidang pengaksesan Internet pada perangkat bergerak, salah satunya Progressive Web App. Progressive Web App adalah Mobile Web yang menggunakan teknologi-teknologi terbaru guna menghasilkan Mobile Web yang memiliki pengalaman pengguna yang jauh lebih baik dari Mobile Web tradisional. Untuk menghasilkan aplikasi *web* dengan pengalaman pengguna yang baik, tentu tidak lepas dari performa dari aplikasi dan teknologi-teknologi dibaliknya. Berbicara tentang performa, banyak hal yang dapat diperhitungkan, beberapa diantaranya adalah waktu respon, penggunaan memori dan penggunaan media penyimpanan. Pada penelitian ini akan membahas mengenai perbandingan antara Progressive Web Apps dan Mobile Web terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan agar dapat memberikan bahan pertimbangan dan referensi untuk penerapan teknologi yang baru dikenalkan ini. Dari pengujian yang dilakukan pada penelitian ini didapatkan beberapa kesimpulan, yaitu kecepatan dalam memuat data menyesuaikan dengan ukuran berkas dan *cache* yang digunakan serta frekuensi pengaksesan halaman aplikasi. Pada ukuran berkas dan *cache* yang kecil Mobile Web masih lebih unggul dibandingkan dengan Progressive Web Apps, sedangkan pada ukuran berkas dan *cache* yang cukup besar Progressive Web Apps mampu mengungguli Mobile Web. Untuk performa terkait penggunaan memori, Mobile Web menggunakan memori yang lebih sedikit dibandingkan dengan Progressive Web Apps dikarenakan adanya proses tambahan pada Progressive Web Apps (Service Worker). Sedangkan untuk performa terkait penggunaan media penyimpanan, pada Mobile Web tidak menggunakan media penyimpanan sama sekali, sedangkan pada Progressive Web Apps penggunaan media penyimpanan menyesuaikan dengan *cache* yang disimpan pada peramban.

Kata Kunci: *web*, *mobile*, *progressive*, aplikasi, performa

ABSTRACT

Nowadays, many new technologies are emerging in the field of Accessing Internet via mobile devices, one of which is Proressive Web App. It's a Mobile Web that uses the latest technologies to create a Mobile Web with better user experiences than traditional Mobile Web. Creating a web application with such experiences, one thing that can not be forgotten is performances and the technologies behind it. Speaking of performance, many things could be considered, such as responses time, memory usages and storage usages. This study will discuss about comparisons between Progressive Web Apps and Mobile Web related to responses time, memory usages and storages usage in order to provide some insights and references of this newly introduced technology. From the tests conducted in this study, there are some things that can concluded. Response time performance depends on page resources size, caches size and the access frequencies. On a small sized page and small sized cache files, Mobile Web still faster than Progressive Web App, while on large sized page and large cache files Progressive Web App is the winner. In memory usage performance, Mobile web uses memory fewer than Progressive Web Apps due to an additional process that Progressive Web Apps had (Service Worker). In storage usages performance, Mobile web did not use any of storage space, while the storage usage of Progressive Web Apps depends on the size of the caches stored in browser.

Keywords: web, mobile, progressive, application, performance

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR KODE PROGRAM

DAFTAR ISTILAH

Android	Sebuah sistem operasi berbasis Linux yang dirancang untuk perangkat bergerak layar sentuh seperti telepon pintar dan komputer tablet.
Aplikasi Natif	Aplikasi yang dibangun dengan bahasa pemrograman yang spesifik untuk *platform* tertentu.
Background Process	Sebuah proses yang berjalan sendiri/otomatis secara periodik/waktu sebenarnya dan berjalan dibelakang layar dengan sedikit atau tanpa interferensi pengguna.
Browser (Peramban)	Perangkat lunak yang berfungsi untuk menerima dan menyajikan sumber informasi di Internet.
Cache	Sebuah komponen yang menyimpan data-data komputasi yang baru dijalankan atau duplikasi data lain yang berguna untuk menjalankan perintah di masa mendatang dengan lebih cepat.
Downasour	Tampilah halaman peraman ketika tidak terhubung dengan jaringan.
Hardware (Perangkat Keras)	Sebuah sistem yang dibentuk dari komponen utama berupa seperangkat alat fisik yang berhubungan dengan perangkat lunak dan komputer untuk menjalankannya atau mengkoordinasikannya.
Homescreen	Tampilan utama pada perangkat bergerak.
Internet	Sebuah jaringan komputer yang saling terhubung satu dengan lainnya.
Jaringan Komputer	Sebuah jaringan telekomunikasi data digital yang memungkinkan berbagi data/sumber daya di antara komputer yang tersambung.
Media Penyimpanan	Suatu media penyimapan sekunder (*secondary storage*) seperti SD Card, Hard Disk, dsb., yang didalamnya tersimpan data, perintah dan informasi.
Memori	Suatu media penyimpanan sementara yang (RAM) digunakan untuk menangani aplikasi dan data yang sedang digunakan atau berjalan.
Offline	Sebuah kondisi ketika seseorang atau suatu perangkat tidak tersambung dengan jaringan Internet.

Online	Sebuah kondisi ketika seseorang atau suatu perangkat sedang tersambung dengan jaringan Internet.
Pengalaman Pengguna	Cara seseorang merasakan ketika menggunakan sebuah produk, sistem atau jasa.
Proxy Server	Sebuah komputer server atau program komputer yang dapat bertindak sebagai komputer lainnya untuk melakukan *request* terhadap konten dari Internet.
Request	Sebuah proses permintaan akan suatu hal (misalkan data) oleh klien terhadap peladen.
Resources	Hal-hal yang dibutuhkan untuk menjalankan aplikasi.
Responses	Jawaban atas suatu permintaan oleh peladen terhadap klien.
Rupa Huruf (*font*)	Sebuah berkas yang memiliki fungsi untuk mengatur bagaimana rupa dari huruf saat ditampilkan.
Sintaksis	Sebuah pengaturan dan hubungan kata dengan kata atau dengan satuan lain yang lebih besar.
Server (Peladen)	Sebuah sistem komputer yang menyediakan jenis layanan tertentu dalam sebuah jaringan komputer.
Software (Perangkat Lunak)	Sebuah sistem yang dibentuk dari komponen utama berupa seperangkat instruksi dan data yang bisa dieksekusi dalam satu atau lebih komputer yang bisa melibatkan satu atau lebih perangkat keras yang lain.
Waktu Respon	Total waktu yang dihabiskan untuk berinteraksi dengan peladen dan memuat seluruh berkas gambar, JavaScript, CSS dan berkas eksternal lainnya
Web	Sebuah halaman informasi yang disediakan melalui jaringan Internet sehingga bisa diakses di seluruh dunia selama terkoneksi dengan jaringan Internet.

BAB 1 PENDAHULUAN

1.1 Latar belakang

Dewasa ini, Internet adalah fenomena global yang terus tumbuh dan menjadi primadona bagi orang-orang diseluruh dunia. Pada tahun 2015 saja, pengguna internet di dunia sudah mencapai angka 3,2 miliar orang. Angka ini jauh berkembang apabila dibandingkan dengan jumlah pengguna Internet pada tahun 2000 silam yang hanya berkisar 400 juta orang (ITU, 2015). Berkembangnya pengguna internet ini tidak lepas dari semakin mudahnya orang-orang dalam mengakses internet. Salah satu hal yang memudahkan orang-orang dalam mengakses internet adalah makin banyaknya perangkat yang mampu terhubung dengan internet, salah satunya adalah perangkat bergerak. Persentase penetrasi internet dari perangkat bergerak di dunia terus meningkat tiap tahunnya. Pada tahun 2016 saja, persentase penetrasi internet dari perangkat bergerak sudah melampaui penetrasi internet dari perangkat lainnya, yakni sebesar 56.1% dari seluruh penetrasi internet didunia dan diproyeksikan akan mencapai 63.4% pada tahun 2019 mendatang (Liu, 2016).

Selain peningkatan penetrasi pengguna, varian teknologi yang mendukung penggunaan internet di perangkat bergerak pun bermunculan. Salah satunya adalah teknologi dalam hal performa dan pengalaman pengguna yang dikenalkan Google pada tahun 2016 silam, Progressive Web Apps. Dijelaskan Rahul Roy-Chowdury, direktur Produk Managemen Google Chrome, Google pada gelaran Google I/O 2016, teknologi Progressive Web Apps adalah Mobile Web yang menghadirkan pengalaman yang lebih baik dengan tampilan dan fitur yang hampir sama dengan Mobile Native Apps (Roy-Chowdury, 2016). Progressive Web Apps tidak dikemas dan dipublikasikan lewat App Store, Progressive Web Apps hanyalah Mobile Web yang diberi kelebihan-kelebihan yang mampu membuat Mobile Web menjadi *top-level* (level atas) pada *task switcher*, *homescreen* dan *notification tray* tanpa harus instalasi diawal. Mobile Web yang bisa mengirimkan notifikasi dan menggunakan fitur-fitur yang ada pada perangkat bergerak pengguna dari waktu ke waktu selama pengguna masih terus menggunakan atau mengunjungi Mobile Web tersebut, sehingga *mobile* (perangkat bergerak) *web* tersebut lama-lama menjadi “aplikasi” (Russell, 2015).

Progressive Web Apps mengedepankan pengalaman pengguna yang lebih baik dibandingkan Mobile Web tradisional (Archibald, 2015). Progressive Web Apps mengangkat tiga parameter utamanya, yaitu *Reliable*, *Fast* dan *Engaging*. *Reliable* artinya Mobile Web yang mampu memuat halaman secara instan dan tidak akan menampilkan *downasaur*, bahkan pada jaringan yang tidak menentu. *Fast* artinya Mobile Web dapat merespon interaksi pengguna secara cepat dengan animasi yang halus. Sedangkan *engaging* artinya Mobile Web terasa seperti Mobile Native Apps pada perangkat bergerak dengan pengalaman pengguna yang mendalam (Google Developers, 2017). Untuk mencapai hal tersebut Progressive Web Apps menggunakan beberapa pendekatan, mulai dari pendekatan dari

pengoptimisasian aplikasi, penggunaan teknologi-teknologi anyar, hingga standarisasi Mobile Web. Hal ini diklaim mampu meningkatkan performa guna meningkatkan kualitas pengalaman pengguna.

Berbicara tentang performa aplikasi tentu banyak hal yang harus diperhitungkan. Salah satunya performa terhadap persepsi pengguna itu sendiri, dimana performa terhadap persepsi pengguna ditentukan berdasarkan parameter-parameter berupa tingkat responsif aplikasi, seberapa cepat aplikasi dimuat, seberapa baik aplikasi dalam menggunakan perangkat keras seperti memori, media penyimpanan dan konsumsi daya, seberapa besar *frame fate* dari aplikasi dan lain sebagainya (AppDynamics, 2014). Dari parameter tersebut, terdapat tiga parameter yang diunggulkan dan bersinggungan dengan teknologi-teknologi dibalik Progressive Web Apps, yaitu waktu respon, penggunaan memori dan penggunaan media penyimpanan.

Waktu respon sistem adalah waktu jeda antara inisiasi pengguna dengan tampilan hasil secara keseluruhan oleh sistem (Miller, 1958). Waktu respon yang disebutkan pada tulisan ini adalah waktu yang diperlukan aplikasi *web* dalam menampilkan keseluruhan isi dari halaman-halaman *web* tersebut. Permasalahan waktu respon yang lambat adalah permasalahan klasik yang dihadapi oleh pengembang aplikasi-aplikasi di dunia, dibuktikan oleh data yang dilansir oleh Akamai Technology (2016) menunjukkan rata-rata kecepatan respon aplikasi *web* yang masih berkisar di angka 3000 milidetik (mdtk) hingga 4000 mdtk pada aplikasi di perangkat komputer dan 5000 mdtk hingga 6000 mdtk pada aplikasi di perangkat bergerak (Belson, 2016). Padahal menurut penelitian-penelitian yang dilakukan sebelumnya, salah satunya yang dilakukan oleh Fiona Fui dan Hoon Nah (2004) bahwa aplikasi yang baik adalah aplikasi yang waktu responnya kurang dari dua detik (*2 seconds rule*). Selain itu, lambatnya waktu respon apabila dilihat dari kacamata bisnis, akan mempengaruhi kuantitas dan frekuensi penggunaan dari aplikasi, mengakibatkan berkurangnya penggunaan dari aplikasi tersebut (Akamai Technologies, 2009).

Mengatasi permasalahan ini Progressive Web Apps menawarkan beberapa solusi, seperti optimisasi konten, penggunaan arsitektur *app shell* dan lain sebagainya. Dengan solusi-solusi tersebut, Progressive Web Apps akan bisa berjalan langsung dari *homescreen* secara cepat dan bisa memuat halaman *web* secara instan dalam kondisi jaringan seperti apapun. Hal ini bisa dilakukan berkat Web App Manifest dan Service Worker. Ada hal yang perlu diperhatikan dari solusi-solusi tersebut, yaitu penggunaan memori, contoh saja cara kerja dari Service Worker yang terpisah dari halaman *web* dikarenakan Service Worker adalah proses yang berjalan di balik layar sehingga perlu diketahui seberapa besar penggunaan memori dalam Progressive Web Apps dibandingkan dengan Mobile Web tradisional. Mengingat penggunaan memori adalah salah satu faktor penentu baik buruknya performa sebuah aplikasi pada perangkat bergerak (AppDynamics, 2014). Selain penggunaan memori, ada hal lain yang perlu diperhatikan yaitu penggunaan media penyimpanan pada perangkat pengguna. Karena, mampu berjalannya Progressive Web Apps dengan cepat dalam kondisi jaringan seperti

apapun berkat disimpannya data-data aplikasi kedalam perangkat pengguna. Padahal, selain terbatasnya media penyimpanan di tiap-tiap perangkat, *browser* (peramban) sebagai aplikasi yang menjalankan *mobile web* memiliki kuota maksimal penggunaan media penyimpanan lokal (Local Storage) (Osmani, 2016) dan berhak untuk menghapus data-data yang tersimpan apabila aplikasi peramban dalam keadaan kuota penyimpanan yang hampir habis (*low storage preasure*) (Archibald, 2014). Oleh karena itu dirasa perlu untuk mengetahui seberapa besar kebutuhan media penyimpanan oleh Progressive Web Apps dibandingkan dengan Mobile Web tradisional.

Dengan adanya permasalahan-peramasalahan tersebut, perlu dilakukan penelitian untuk mengetahui perbedaan performa Progressive Web Apps dan Mobile Web terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan. Mengingat, pentingnya performa aplikasi terhadap pengalaman pengguna yang lebih baik. Selain itu, dengan diketahui perbandingan tersebut dapat diketahui bagaimana performa Progressive Web Apps dan Mobile Web waktu respon, penggunaan memori dan penggunaan media penyimpanan.

Dengan mengetahui perbandingan performa Progressive Web Apps dan Mobile Web, diharapkan dapat dijadikan sebagai bahan pertimbangan untuk penerapan teknologi Progressive Web Apps ataupun Mobile Web agar mengurangi permasalahan-permasalahan yang dialami pengembang dan pengguna aplikasi pada perangkat bergerak. Selain itu, hasil perbandingan ini dapat dijadikan sebagai bahan referensi untuk para peneliti dalam mengembangkan dan mendalami Progressive Web Apps pada perangkat bergerak.

1.2 Rumusan masalah

Berdasarkan latar belakang diatas, maka rumusan masalah yang dapat dirujuk adalah sebagai berikut:

1. Bagaimana perbandingan performa Progressive Web Apps dan Mobile Web terkait waktu respon terhadap proses memuat dan menampilkan halaman aplikasi?
2. Bagaimana perbandingan performa Progressive Web Apps dan Mobile Web terkait terkait penggunaan memori terhadap proses menjalankan aplikasi?
3. Bagaimana perbandingan performa Progressive Web Apps dan Mobile Web terkait penggunaan media penyimpanan terhadap seluruh berkas yang disimpan oleh aplikasi pada perangkat bergerak?

1.3 Tujuan

Adapun tujuan yang ingin dicapai oleh peneliti, antara lain:

1. Mengetahui perbandingan performa Progressive Web Apps dan Mobile Web terkait terkait waktu respon terhadap proses memuat berkas-berkas untuk menampilkan halaman aplikasi.

2. Mengetahui perbandingan performa Progressive Web Apps dan Mobile Web terkait penggunaan memori terhadap proses menjalankan aplikasi.
3. Mengetahui perbandingan performa Progressive Web Apps dan Mobile Web terkait penggunaan media penyimpanan terhadap seluruh berkas yang disimpan oleh aplikasi pada perangkat bergerak.

1.4 Manfaat

Penelitian ini diharapkan memiliki manfaat yang baik dan berguna bagi pembaca maupun orang-orang. Adapun manfaat yang diharapkan adalah sebagai berikut:

1. Memberikan bahan pertimbangan untuk penerapan teknologi Progressive Web Apps agar mengurangi permasalahan-permasalahan yang dialami pengembang dan pengguna aplikasi pada perangkat bergerak.
2. Dapat dijadikan sebagai bahan referensi untuk para peneliti dalam mengembangkan dan mendalami Progressive Web Apps pada perangkat bergerak.

1.5 Batasan masalah

Agar penelitian yang dilakukan tidak terlalu luas, maka perlu adanya batasan-batasan dari permasalahan dalam penelitian ini. Adapun batasan masalah adalah sebagai berikut:

1. Berkas-berkas yang digunakan berupa berkas HTML, CSS, JavaScript, JSON, teks, gambar dan video.
2. Mobile Web tradisonal adalah Mobile Web yang tidak menggunakan teknologi khusus yang digunakan dalam Progressive Web Apps ataupun teknologi-teknologi lainnya seperti Accelerated Mobile Pages, AppCache dan lain sebagainya.

1.6 Sistematika pembahasan

Sistematika penulisan dalam skripsi ini sebagai berikut:

BAB 1 PENDAHULUAN

Memuat latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Membahas tentang teori dan referensi yang mendasari analisis kinerja Progressive Web Apps terkait waktu respon, ukuran transfer data dan besar penggunaan media penyimpanan pada perangkat bergerak.

BAB 3 METODOLOGI

Membahas metode dan alur yang digunakan dalam pengujian, dimulai dari studi literatur, analisis kebutuhan dan skenario pengujian.

BAB 4 PEMBAHASAN

Membahas pengujian yang akan dilakukan secara lebih rinci.

BAB 5 ANALISIS

Menguraikan dan menganalisis hasil yang didapat dari pengujian sesuai skenario pengujian.

BAB 6 PENUTUP

Memuat kesimpulan yang diperoleh dari pengujian dan analisis, serta saran-saran untuk pengembangan lebih lanjut.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Mobile Web

Secara teori, Mobile Web hampir sama dengan situs web yang lain. Mobile Web menggunakan HTML berbasis peramban yang bisa diakses oleh perangkat bergerak. Tidak seperti situs web yang dibuat untuk diakses melalui komputer *desktop*, Mobile Web didesain untuk dilihat pada tampilan yang lebih kecil. Mobile Web adalah versi dari situs web biasa yang secara spesifik digunakan pada perangkat bergerak (BlueFountainMedia, 2015). Umumnya, Mobile Web dibentuk dari tiga jenis tipe berkas, yaitu HTML, CSS dan JavaScript.

2.1.1 HTML

HTML adalah singkatan dari *HyperText Markup Language*, yang biasa digunakan sebagai bahasa atau kode standar untuk menampilkan sebuah halaman *web* pada aplikasi peramban (Nurhaman, 2016). HTML berfungsi untuk menampilkan informasi-informasi pada situs *web* dengan format yang sederhana yang nantinya akan diterjemahkan oleh peramban ke dalam bentuk yang mudah dimengerti manusia. Contoh cuplikan kode program dari sebuah berkas HTML pada Mobile Web dapat dilihat dalam Kode Program 2.1.

```
 1  <!DOCTYPE html>
 2  <html>
 3  <head>
 4     <meta name="viewport" content="width=device-width, initial-
    scale=1.0">
 5       <title>Hello World!</title>
 6  </head>
 7  <body>
 8     <p>Hello World</p>
 9  </body>
10  </html>
```

Kode Program 2.1 Contoh Sintaksis HTML

Pada baris ke-1 dan ke-2 berfungsi untuk mendefinisikan bahwa berkas tersebut adalah berkas dengan tipe HTML. Pada baris ke-3 adalah inisiasi dari blok Header dari halaman HTML, yang mana diakhiri pada baris ke-6. Pada baris ke-4 dibutuhkan agar perambam mengkompilasi halaman *web* dalam *viewport* yang umumnya digunakan pada perangkat bergerak. Baris ke-5 adalah pendefinisian judul dari halaman *web*. Baris ke-7 hingga ke-9 adalah konten dari halaman *web* tersebut.

2.1.2 CSS

CSS (*Cascading Style Sheets*) adalah mekanisme untuk menambahkan corak pada dokumen *web*, seperti mengatur tata letak, warna, efek, *font* dan lain sebagainya (W3, 1996). Hingga saat ini sudah terdapat tiga versi dari CSS. Pada versi pertama (CSS1), CSS dikembangkan berpusat pada pemformatan *style* dari dokumen HTML, Sedangkan pada versi ke dua (CSS2) dikembangkan agar bisa menyisipkan *style* dari dokumen secara terpisah. Lalu pada versi terbaru yaitu

versi ketiga (CSS3), CSS dikembangkan untuk menambah fitur-fitur seperti animasi, *media-query*, transformasi dan lain sebagainya. Contoh cuplikan kode program dari sebuah berkas CSS pada Mobile Web dapat dilihat dalam Kode Program 2.2.

```
1  body {
2    font-family: 'Helvetica';
3    font-size: 12px;
4    color: #000;
5    background: #FFF;
6  }
```

Kode Program 2.2 Contoh Sintaksis CSS

Pada Kode Program 2.2 adalah cuplikan dari sintaksis CSS yang berfungsi untuk memberikan corak pada bagian `body` halaman *web*. Corak yang diberikan adalah bentuk huruf menjadi Halvetica yang ditulis pada baris ke-2, menentukan ukuran huruf pada baris ke 3, memberikan warna pada huruf dibaris ke-4 dan memberikan warna latar pada baris ke-5.

2.1.3 JavaScript

JavaScript adalah bahasa pemrograman yang umumnya digunakan sebagai bahasa pemrograman untuk halaman *web* pada sisi klien (Mozilla, 2016). JavaScript memungkinkan untuk sebuah halaman *web* dapat berinteraksi dengan pengguna secara langsung tanpa harus memuat ulang kembali halaman *web* tersebut. Contoh ketika ada sebuah tombol yang berfungsi untuk menampilkan sebuah pesan tersembunyi ketika ditekan; dengan JavaScript, halaman tersebut tidak perlu memuat ulang untuk menampilkan pesan tersebut. Cukup memanggil fungsi yang sudah ada didefinisikan pada berkas JavaScript. Contoh cuplikan kode program dari sebuah berkas JavaScript pada Mobile Web dapat dilihat dalam Kode Program 2.3.

```
1  var object = document. getElementById('tombol');
2  object.addEventListener("click", myScript);
3  myScript = function(){
4    alert(“Ini pesan tersembunyi”);
5  }
```

Kode Program 2.3 Contoh Sintaksis JavaScript

Pada Kode Program 2.3, baris ke 1 adalah menginisiasi variabel `object` dengan DOM yang memiliki atribut `id` “tombol”. Lalu pada baris ke-2, `object` disisipkan fungsi `myScript` yang akan berjalan ketika `object` tersebut ditekan. Pada baris ke-3 sampai ke-5 adalah sintaksis yang mendefinisikan isi dari fungsi `myScript`.

Selain beberapa kegunaan dari JavaScript yang sudah dijelaskan, masih banyak lagi kegunaan-kegunaan yang dimiliki JavaScript. Kegunaan-kegunaan ini sungguh penting dalam pembuatan Mobile Web. Kegunaan-kegunaan lainnya seperti membuat *request* data pada peladen dari proses *background* aplikasi, validasi masukan data dari pengguna, memperbarui isi konten dan lain sebagainya.

2.2 Progressive Web Apps

Progressive Web Apps (Aplikasi Web Progresif) adalah teknologi yang menggabungkan fitur terbaik dari *web* dan aplikasi perangkat bergerak (LePage, 2016). Progressive Web Apps tidak perlu adanya instalasi karena pengguna membangun hubungan dengan aplikasi terus menerus sehingga akan semakin canggih dari waktu ke waktu. Canggih dalam arti Progressive Web Apps akan dapat menyimpan fungsional-fungsional yang ada pada aplikasi seiring diaksesnya halaman-halaman web yang terdapat pada aplikasi tersebut. Sehingga kedepannya Progressive Web Apps dapat berjalan dengan lancar dalam kondisi jaringan apapun, termasuk pada kondisi luring seperti layaknya aplikasi natif pada perangkat bergerak. Selain itu, walaupun Progressive Web Apps berbasis Mobile Web App, Progressive Web App tetap bisa melakukan beberapa hal yang tidak dapat dilakukan oleh Mobile Web App seperti Push Notifications, memiliki ikon pada tampilan utama perangkat bergerak (*homescreen*), memiliki pengalaman pengguna seperti menggunakan aplikasi natif dan fitur-fitur lainnya yang disediakan oleh teknologi *web* saat ini. Oleh karena itu dapat disimpulkan bahwa Progressive Web Apps adalah cara untuk membuat sebuah Mobile Web yang memiliki pengalaman pengguna yang semirip mungkin dengan aplikasi natif (Stauffler, 2017). Untuk mencapai hal tersebut, Progressive Web Apps menggunakan beberapa teknologi, diantaranya Service Worker, Application Shell dan Web App Manifest.

Perlu diketahui bahwa pengelolaan *cache* pada Service Worker berbeda dengan pengelolaan *cache* pada peramban. Dimana pada peramban pengelolaan *cache* dilakukan oleh Cache-Control peramban. Cache-Control pada peramban menentukan berkas-berkas yang mana yang akan disimpan berdasarkan HTTP Response Headers (Krauss, 2016). HTTP Response Headers sendiri merupakan sebuah informasi mengenai bagaimana pengelolaan *cache* pada aplikasi secara global, tidak spesifik seperti pada Service Worker. Selain itu, *cache* yang disimpan melalui HTTP Response Headers juga memiliki tenggang waktu (*timeout*) sampai kapan berkas tersebut dapat disimpan oleh peramban.

2.2.1 Service Worker

Service Worker adalah program yang berjalan dibalik layar, independen terhadap aplikasi dan berjalan untuk mengatur *network request*, *Push Notification*, perubahan konektifitas dan lain sebagainya (Lynch, 2016). Service Worker akan berjalan terus pada peramban hingga pengguna memberhentikan atau menghapusnya melalui pengaturan pada peramban. Pada dasarnya, Service Worker adalah sebuah berkas JavaScript yang dimiliki oleh sebuah aplikasi yang berjalan pada proses yang berbeda, yang berarti Service Worker tidak akan berhenti ketika pengguna menutup aplikasi *web* (bahkan aplikasi peramban sendiri) (Semenov, 2017).

Pada pengimplementasiannya, umumnya terdapat tiga fungsi utama atau tahapan yang dimiliki oleh Service Worker, yaitu *install* untuk memasang Service

Worker pada peramban, lalu *activate* untuk mengaktifasi Service Worker yang sudah terpasang pada peramban dan *fetch* untuk mendapatkan berkas-berkas yang disimpai sebagai *cache* disesuaikan dengan *request* dari halaman web tersebut.

Dalam Kode Program 2.4 adalah contoh kode Service Worker pada tahapan *install*. Fungsi dari kode tersebut adalah mendefinisikan sebuah fungsi untuk memasang Service Worker yang dapat dilihat pada baris ke-1 dan berakhir pada baris ke-7. Adapun isi dari fungsi tersebut, yang pertama adalah menggunakan Promise pada JavaScript pada baris ke-2 dan baris ke-3. Fungsi dari Promise sendiri adalah untuk merepresentasikan *event* dari penyelesaian (atau kegagalan) sebuah operasi yang *asynchronous* dan nilai yang dihasilkan (Mozilla, 2017). Selain itu, pada baris ke-3 terdapat fungsi untuk meregistrasi atau membuka *cache* dengan nama isi dari variabel `cacheName`. Setelah *cache* diregistrasi, selanjutnya pada baris ke-4 adalah menambahkan berkas-berkas yang ingin disimpan pada *cache* yang yang diregistrasi tadi.

```
1  self.addEventListener('install', function(e) {
2    e.waitUntil(
3      caches.open(cacheName).then(function(cache) {
4        return cache.addAll(filesToCache);
5      })
6    );
7  });
```

Kode Program 2.4 Contoh Kode Install pada Service Worker

Tentu isi dari tahapan *install* masih bisa dimodifikasi lagi agar dapat menyesuaikan kebutuhan dari aplikasi yang dikembangkan, misalnya menambahkan sebuah konfirmasi untuk mengetahui sukses atau tidaknya proses menambahkan *cache*, dan lain sebagainya.

```
1  self.addEventListener('activate', function(e) {
2    e.waitUntil(
3      caches.keys().then(function(keyList) {
4        return Promise.all(keyList.map(function(key) {
5          if (key !== cacheName && key !== dataCacheName) {
6            return caches.delete(key);
7          }
8        }));
9      })
10   );
11   return self.clients.claim();
12 });
```

Kode Program 2.5 Contoh Kode Activate pada Service Worker

Dalam Kode Program 2.5 adalah contoh kode Service Worker pada tahapan *activate*. Secara garis besar, fungsi dari potongan kode tersebut adalah memperbarui berkas *cache* yang lama dengan yang baru apabila terjadi perubahan berkas yang diterima dari peladen. Pada baris ke-3 digunakan untuk melihat daftar berkas yang dimiliki oleh aplikasi yang terdaftar pada *cache* peramban. Dari daftar tersebut, yang selanjutnya dilakukan adalah melakukan pengecekan *cache* yang saat ini tersimpan dengan *cache* yang ingin disimpan. Apabila terdapat ketidaksesuaian, maka berkas *cache* tersebut akan dihapus. Inti

dari tahapan ini adalah untuk menjamin kesesuaian berkas pada peramban dengan berkas yang ada pada peladen.

```
1  self.addEventListener('fetch', function(e) {
2    e.respondWith(
3      caches.match(e.request).then(function(response) {
4        if(response) return response;
5        else return fetch(e.request)
6      })
7    );
8  });
```

Kode Program 2.6 Contoh Kode Fetch pada Service Worker

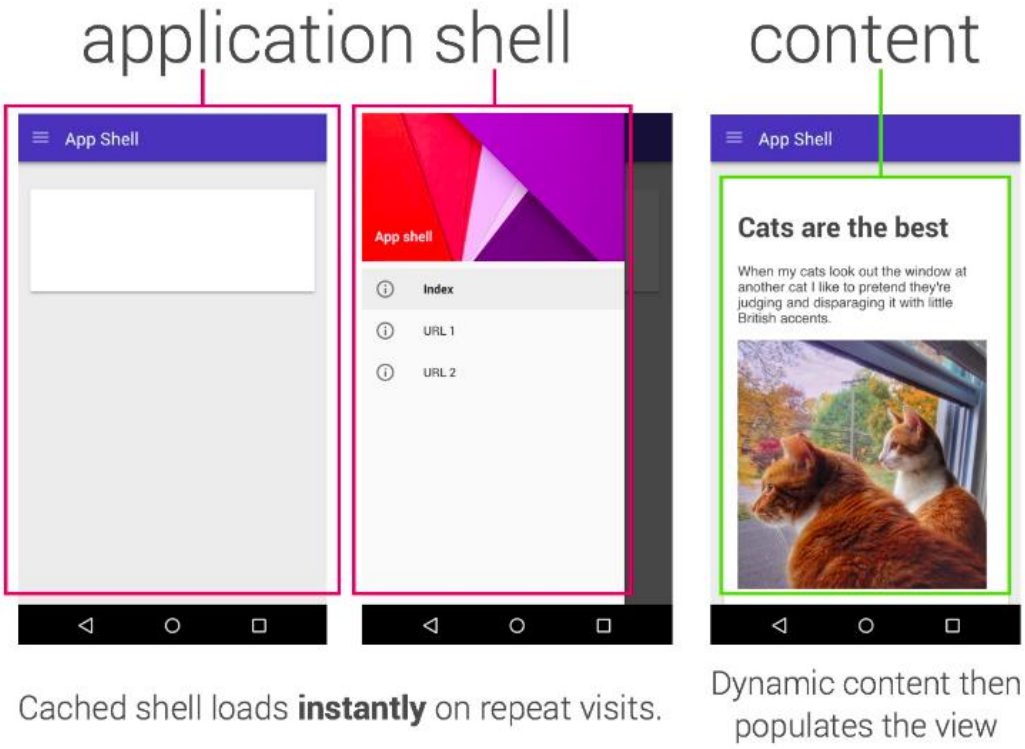
Dalam Kode Program 2.6 adalah contoh kode Service Worker pada tahapan *fetch*. Secara garis besar, fungsi dari potongan kode tersebut dibagi menjadi 2 tahapan, tahapan pertama adalah mengecek apabila terdapat kecocokan antara *request* yang dilakukan dengan berkas yang ada pada *cache*. Apabila terdapat kecocokan maka Service Worker akan memberikan *response* dari *cache* terlebih dahulu lalu, kemudian melanjutkan request tersebut ke peladen. Tahapan kedua adalah menyimpan *response* dari peladen ketika request selesai dan menampilkan pada halaman aplikasi. Adapun proses pengecekan dapat dilihat pada baris ke-3. Lalu pada baris ke-4 adalah pengecekan kondisi dimana peladen berhasil atau tidaknya dalam memberikan *response*. Apabila berhasil maka fungsi ini akan memberikan isi dari respose tersebut kepada aplikasi yang dapat dilihat pada baris ke 5. Apabila gagal makan fungsi ini akan memberikan berkas yang tersimpan pada *cache*.

2.2.2 Application Shell

Application Shell adalah bentuk minimal dari HTML, CSS dan JavaScript yang dibutuhkan untuk menampilkan tampilan utama dari sebuah aplikasi. Ketika Progressive Web Apps pertama kali diakses, Progressive Web Apps dengan segera akan menyimpan berkas-berkas yang dibutuhkan oleh Application Shell. Sehingga, apabila aplikasi tersebut diakses kembali, Progressive Web Apps akan menyediakan seluruh berkas yang dibutuhkan oleh Application Shell langsung dari *cache* peramban, oleh karena itu hal ini mempercepat waktu yang dibutuhkan oleh pengguna untuk melihat antarmuka pada layar perangkat pengguna (Semenov, 2017).

Dalam pengimplementasian Application Shell, tidak terdapat metode-metode khusus atau sintaksis-sintaksis khusus yang dibubuhi pada berkas HTML, CSS dan JavaScript. Namun terdapat beberapa persyaratan yang idealnya harus dipenuhi. Secara Ideal, Application Shell harus dapat dimuat dengan cepat, menggunakan data sedikit mungkin, menggunakan berkas statis dari *cache*, memisahkan konten dan navigasi, menerima dan menampilkan konten yang spesifik seperti HTML, JSON dan lain sebagainya serta menyimpan *cache* konten yang bersifat dinamis secara opsional (Osmani, 2017). Adapun contoh tampilan dan pembagian dari Application Shell dapat dilihat dalam Gambar 2.1. Pada bagian kiri adalah implementasi dari Application Shell, yang mana dibagian tersebut terdapat

kerangka konten dari suatu halaman (sebelah kiri) dan navigasi halaman (bagian tengah). Sedangkan pada bagian kanan adalah contoh Application Shell yang sudah terisi konten suatu halaman.



Gambar 2.1 Contoh Tampilan dan Bagian pada Application Shell

Sumber: Osmani (2017)

2.2.3 Web App Manifest

Web App Manifest adalah berkas JSON sederhana yang memberikan kemampuan kepada pengembang untuk mengatur bagaimana aplikasi terlihat oleh pengguna pada area dimana pengguna akan diharapkan untuk melihat aplikasi tersebut (misalkan pada *homescreen* perangkat bergerak), menjalankan aplikasi oleh pengguna secara langsung serta mendefinisikan bagaimana tampilan aplikasi ketika dijalankan. Ketika sebuah situs dengan Web App Manifest dijalankan, terdapat beberapa keunggulan. Diantaranya adalah situs *web* tersebut dapat gambar *icon* yang unik sehingga pengguna dapat membedakan situs tersebut dengan situs yang lain, dapat menampilkan sesuatu ketika *resources* dimuat atau dimuat ulang dari berkas *cache* dan dapat menentukan karakteristik *default* kepada peramban untuk menghindari transisi yang aneh ketika *resources* tersedia (selesai dimuat). Semua hal ini dapat dilakukan dengan mekanisme sederhana pada sebuah berkas teks, yaitu Web App Manifest (Gaunt, 2017). Adapun contoh berkas JSON dalam pengimplementasian dari Web App Manifest dapat dilihat dalam Kode Program 2.7.

```
 1  {
 2    "name": "Nama Aplikasi",
 3    "icons": [{
 4        "src": "images/favicon.png",
 5        "sizes": "144x144",
 6        "type": "image/png"
 7      }],
 8    "start_url": "index.html",
 9    "display": "standalone",
10    "background_color": "#FFFFFF",
11    "theme_color": "#FFFFFF"
12  }
13
```

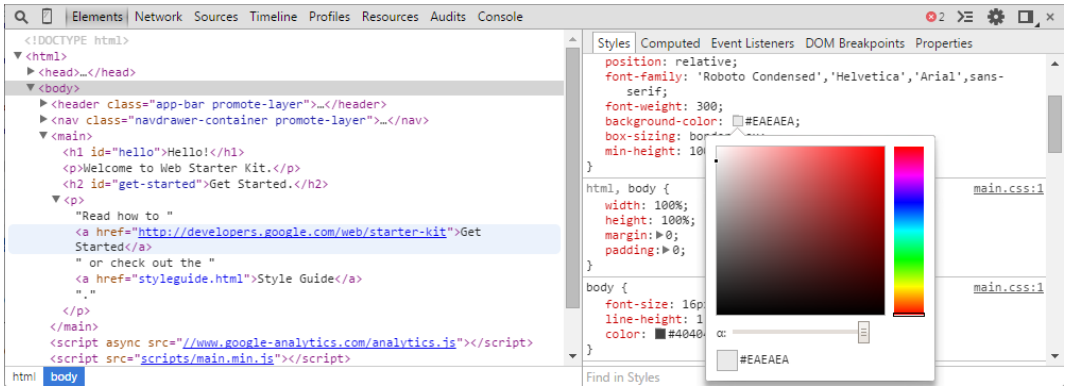
Kode Program 2.7 Contoh Web App Manifest

Dalam Kode Program 2.7 adalah contoh Web App Manifest yang memberikan definisi pada suatu aplikasi. Adapun definisi yang diberikan pada contoh tersebut adalah nama aplikasi, ikon, halaman awal, tipe tampilan, warna latar dan warna tema. Nama aplikasi yang ditulis pada baris ke-2 adalah tulisan yang muncul dibawah gambar ikon yang muncul pada *homescreen* perangkat bergerak. Ikon yang ditulis pada baris ke-3 hingga ke-7 adalah definisi lengkap dari gambar ikon berupa letak berkas gambar, ukuran gambar dan tipe gambar. Halaman awal yang ditulis pada baris ke-8 adalah definisi halaman mana yang dituju ketika aplikasi pertama kali diluncurkan. Tipe tampilan sendiri adalah opsi yang diberikan peramban untuk menampilkan halaman aplikasi. Terdapat dua tipe untuk menampilkan halaman, yaitu *standalone* yang menyembunyikan *User Interface* (UI) peramban dan *browser* yang menampilkan UI peramban layaknya situs *web* biasanya. Adapun cara penulisan dapat dilihat pada baris ke-9. Pada baris ke-10 dan ke-11 adalah contoh penulisan untuk pendefinisian warna latar dan warna tema aplikasi.

2.3 Chrome Developer Tools

Chrome Developer Tools (Chrome DevTools) adalah kumpulan alat *authoring* dan *debugging web* yang terpasang pada peramban Google Chrome. Chrome DevTools menyediakan akses yang mendalam kepada pengembang *web* pada internal peramban dan aplikasi *web* mereka. Chrome DevTools digunakan untuk melacak isu pada *layout*, kumpulan *breakpoint* JavaScript dan mendapat wawasan untuk mengoptimasi kode program (Chrome Developer, n.d). Adapun tampilan antarmuka dari Chrome DevTools dapat dilihat dalam Gambar 2.2. Secara garis besar, terdapat delapan kelompok utama alat yang tersedia pada Chrome DevTools. Beberapa diantaranya adalah Network, Performance, Application dan Memory. Selain itu terdapat pula beberapa fitur lainnya, salah satunya adalah Remote Devices.

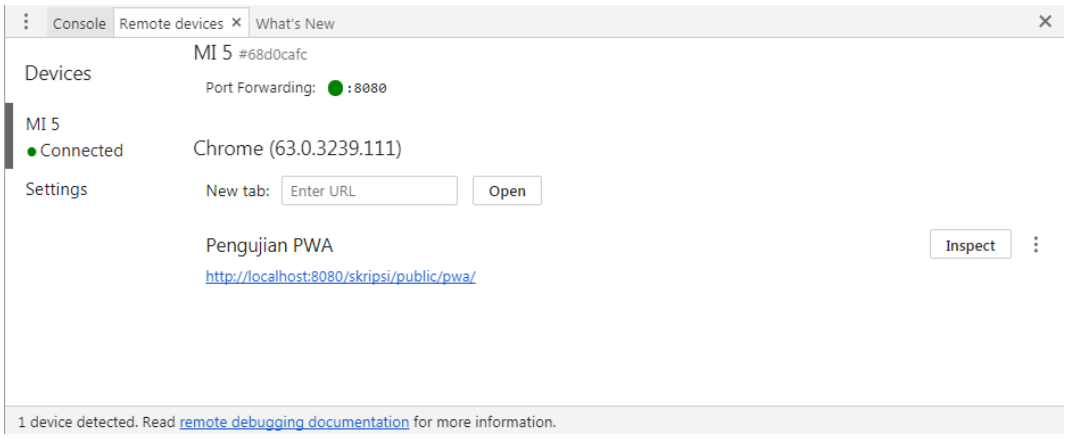
Salah satu kegunaan dari Remote Devices adalah diberikannya akses untuk *authoring* dan *debugging web* yang berjalan pada perangkat bergerak dari perangkat pengembangan, contohnya komputer.



Gambar 2.2 Tampilan Jendela Chrome DevTools

Sumber: Chrome Developer (n.d)

Untuk melakukan Remote Devices, hal yang pertama dilakukan adalah mengubungkan perangkat bergerak dengan komputer menggunakan kabel USB Connector lalu memilih perangkat yang dituju. Selanjutnya adalah menekan tombol Inspect pada judul halaman yang diinginkan. Setelah itu jendela Chrome Devtools akan muncul dan bisa digunakan. Untuk terhubung dengan perangkat bergerak, terdapat beberapa kebutuhan yang harus dipenuhi terlebih dahulu, yaitu mengaktifkan Developers Mode pada perangkat bergerak dan memasang *driver* USB Android yang sesuai dengan perangkat bergerak yang digunakan. Adapun tampilan antarmuka dari Remote Devices dapat dilihat dalam Gambar 2.3.



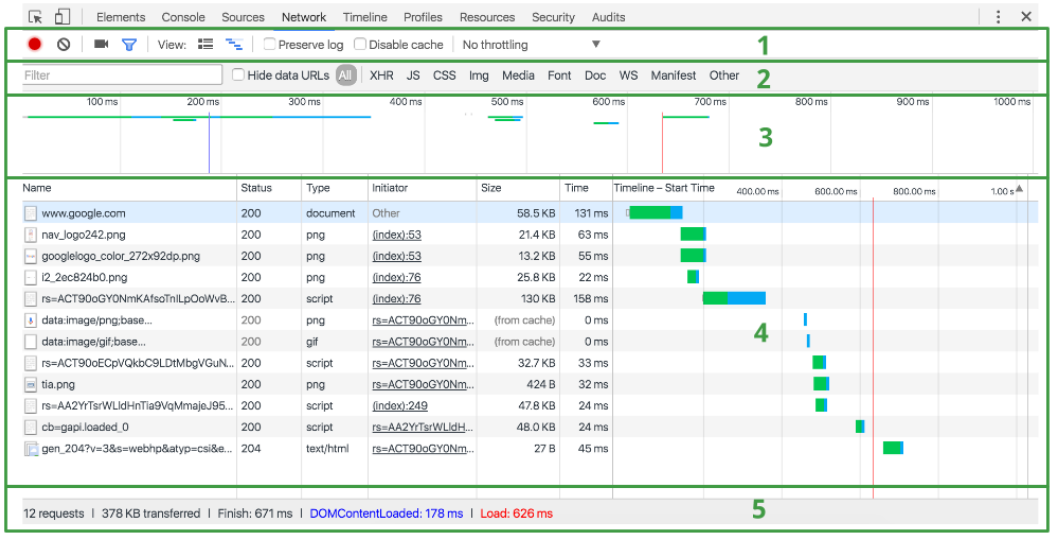
Gambar 2.3 Tampilan Remote Devices

2.3.1 Network

Panel Network menyediakan wawasan terhadap sumber daya (*resources*) yang di-*request* dan diunduh melalui jaringan dalam waktu yang sebenarnya (Basques, 2017). Adapun tampilan antarmuka dari panel Network dapat dilihat dalam Gambar 2.4. Terdapat lima bagian pada panel Network, yaitu:

1. Kontrol. Digunakan untuk mengatur tampilan dan fungsi dari Network.
2. Saringan. Digunakan untuk memilih *resources* mana yang akan ditampilkan pada Tabel Request.

3. Gambaran. Sebuah grafik yang menunjukkan lini masa dari waktu ketika *resources* diterima.
4. Tabel Request. Tabel yang menjabarkan setiap *resources* yang diterima.
5. Kesimpulan. Berisi kilasan dari panel Network yaitu total *requests*, jumlah data yang ditransfer dan waktu muat data.



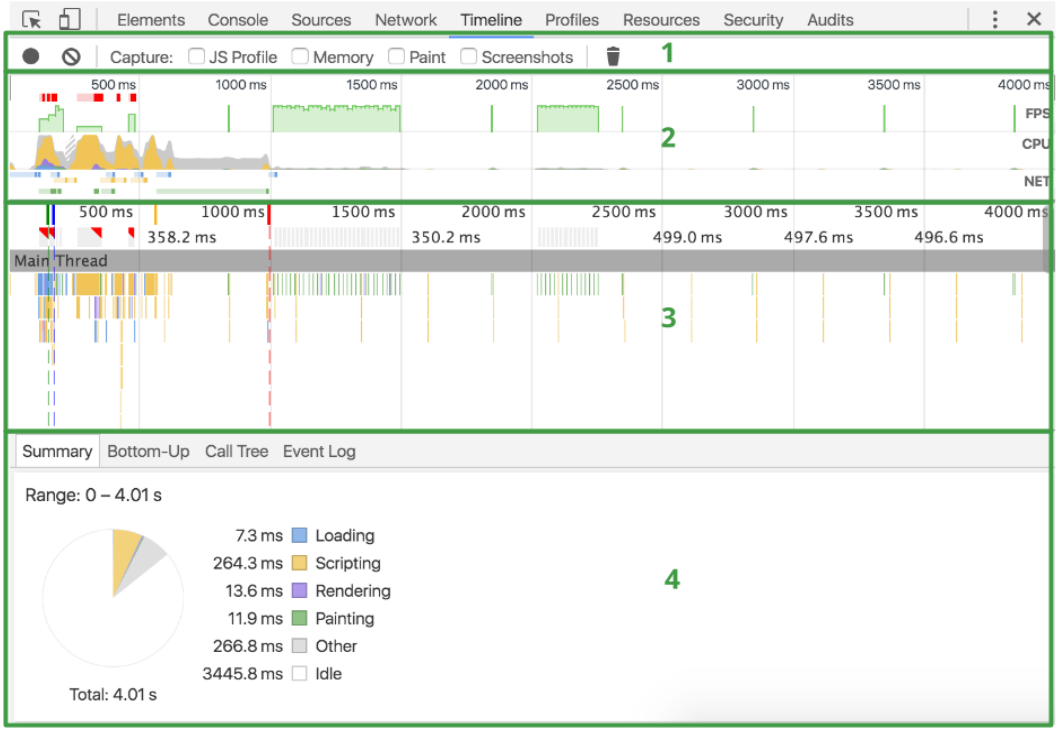
Gambar 2.4 Tampilan Network serta Bagiannya

Sumber: Basques (2017)

2.3.2 Performance

Panel Performance menyediakan gambaran yang lengkap dari waktu yang digunakan ketika memuat dan menggunakan halaman *web* (Basques, 2017). Adapun tampilan antarmuka dari panel Performance dapat dilihat dalam Gambar 2.5. Panel Performance terdiri dari 4 bagian, yaitu:

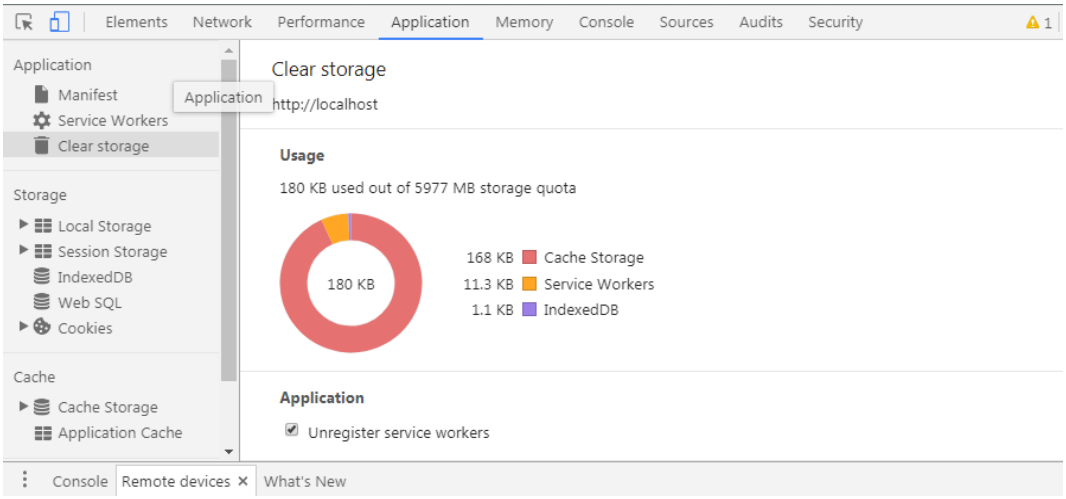
1. Kontrol. Berfungsi untuk memulai dan menghentikan rekaman performa serta mengkonfigurasi informasi apa saja yang ditangkap saat perekaman.
2. Gambaran. Berisi gambaran tingkat tinggi dari performa halaman.
3. Detil Gambaran. Berisi gambaran tinkat tinggi dengan rentang waktu yang dapat dipilih dari bagian Gambaran.
4. Grafik. Visualisasi dari hasil rekaman performa halaman.



Gambar 2.5 Tampilan Panel Performance serta Bagiannya
Sumber: Basques (2017)

2.3.3 Application

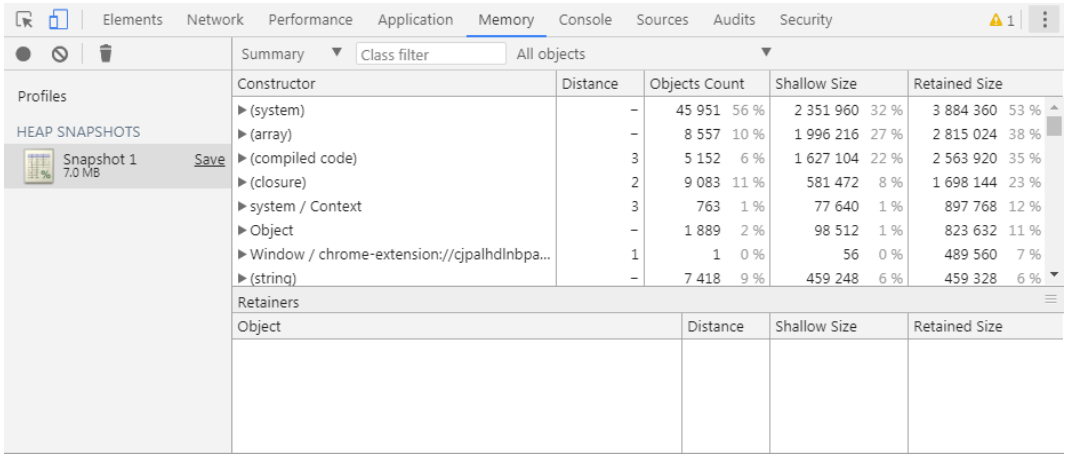
Panel Application digunakan untuk menginspeksi *resources* yang dimuat pada halaman *web*. Pada panel ini juga bisa digunakan untuk berinteraksi dengan basis data HTML5, Local Storage, *cookies*, App*Cache* dan lain sebagainya. Dengan kata lain pada panel ini dapat dilihat seberapa besar media penyimpanan yang digunakan, berkas-berkas yang berada pada media penyimpanan dan lain sebagainya. Adapun tampilan antarmuka dari panel Application dapat dilihat dalam Gambar 2.6.



Gambar 2.6 Tampilan Panel Application

2.3.4 Memory

Panel Memory digunakan untuk mengetahui waktu eksekusi dan penggunaan memori dari halaman *web*. Terdapat tiga fungsi pada panel Memory, salah satunya adalah Take Heap Snapshot. Take Heap Snapshot digunakan untuk melihat distribusi memori dari objek JavaScript dan DOM *nodes* pada suatu halaman *web*. Adapun tampilan antarmuka dari panel Memory dapat dilihat dalam Gambar 2.7.



Gambar 2.7 Tampilan Panel Memory

BAB 3 METODOLOGI

Pada bab ini akan membahas tentang tahapan-tahapan yang digunakan dalam menyelesaikan penelitian ini. Adapun tahapannya terdiri dari studi literatur, analisis kebutuhan, rancangan objek kajian, pengujian, analisis hasil pengujian serta kesimpulan dan saran. Adapun diagram alir pada penelitian ini dapat dilihat dalam Gambar 3.1.

Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Studi literatur

Studi literatur adalah langkah untuk mengumpulkan dan mempelajari literatur dari beberapa bidang ilmu yang berkaitan dalam analisis performa Progressive Web Apps dan Mobile Web terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan. Dengan mengetahui literatur-literatur yang digunakan dalam penelitian ini, akan memudahkan dalam memahami langkah-langkah yang dilakukapan pada penelitian ini. Adapun pustaka studi literatur yang perlu untuk dipelajari untuk melaksanakan penelitian ini adalah sebagai berikut:

a. Mobile Web.
b. Progressive Web Apps.
c. Chrome Developer Tools.

3.2 Analisis kebutuhan

Analisis kebutuhan adalah persiapan dalam menyiapkan bahan-bahan yang akan dibutuhkan untuk melakukan penilitian. Hal ini dibutuhkan agar pengujian ini dapat berjalan dengan lancar tanpa adanya hambatan-hambatan dalam menjalankan, menguji dan melihat hasil dari pengujian. Analisis kebutuhan dalam penelitian ini dibagi menjadi tiga, yaitu analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras dan analisis kebutuhan lingkungan pengujian. Selain itu juga akan disertakan spesifikasi atau versi dari perangkat lunak dan perangkat keras yang digunakan untuk memberikan gambaran yang lebih mendalam terhadap lingkungan saat pengujian dilakukan.

3.2.1 Analisis kebutuhan perangkat lunak

Pengujian pada penelitian ini membutuhkan beberapa perangkat lunak baik pada perangkat komputer dan perangkat bergerak. Perangkat lunak pada perangkat komputer nantinya dibutuhkan untuk menjalankan aplikasi, baik pada bagian peladen maupun dalam hal untuk menguji dan melihat hasil dari pengujian. Sedangkan perangkat lunak pada perangkat bergerak dibutuhkan untuk menjalankan aplikasi yang diuji.

3.2.2 Analisis kebutuhan perangkat keras

Pada penelitian ini membutuhkan beberapa perangkat keras. Perangkat keras ini dibutuhkan untuk menjalankan perangkat lunak yang dibutuhkan untuk menjalankan dan menguji aplikasi yang akan diuji serta untuk menghubungkan perangkat komputer dan perangkat bergerak. Analisis ini didasarkan dari *minimal requirements* (kebutuhan minimal) dari perangkat lunak yang dibutuhkan dalam pengujian ini.

3.2.3 Analisis kebutuhan lingkungan pengujian

Pada penelitian ini membutuhkan beberapa kebutuhan atau konfigurasi tentang lingkungan pengujian. Konfigurasi yang dimaksud seperti kondisi jaringan, kecepatan akses server, pengaturan kecepatan processor dan aktif atau tidaknya pengelolaan *cache* oleh peramban. Hal ini dilakukan agar pengujian dapat menghasilkan data hasil uji yang mudah dibaca atau dianalisis serta data hasil uji yang sejalan dengan pokok permasalahan pada penelitian ini.

3.3 Rancangan Objek Kajian

Rancangan objek kajian membahas bagaimana bentuk antarmuka, struktur berkas, alur pertukaran data dan manajemen *cache* pada objek kajian, dalam hal ini adalah aplikasi dengan teknologi Progressive Web Apps dan Mobile Web. Hal ini dilakukan untuk mempermudah dalam memahami tahapan-tahapan yang ada pada pengujian.

3.3.1 Struktur berkas

Struktur berkas membahas berkas-berkas apa saja yang dibutuhkan oleh masing-masing halaman dan masing-masing tipe aplikasi, mengingat masing-masing halaman memiliki fungsional tersendiri dan masing-masing tipe aplikasi memiliki kebutuhan teknologi yang berbeda. Selain itu juga membahas proporsi ukuran berkas tersebut terhadap ukuran total dari suatu halaman. Hal ini dilakukan untuk membuat aplikasi yang diuji sesuai dengan realita yang ada saat ini. Hasil dari tahapan ini nantinya akan digunakan sebagai acuan dalam merancang antarmuka halaman aplikasi.

3.3.2 Antarmuka

Bagian ini akan membahas bagaimana bentuk dari antarmuka aplikasi. Antarmuka aplikasi dibahas mulai dari bagian-bagian pembangun antarmuka halaman dan blok-blok penyusun konten halaman. Hal ini dilakukan untuk memudahkan dalam mendeskripsikan fungsionalitas dari masing-masing halaman.

Terdapat tiga halaman pada masing-masing aplikasi, yaitu halaman optimal, halaman sedang dan halaman berat. Masing-masing tipe aplikasi memiliki antarmuka yang sama karena pengujian dilakukan dengan membandingkan dua tipe aplikasi serupa dengan teknologi-teknologi yang berbeda sesuai dengan kebutuhan masing-masing tipe aplikasi.

3.4 Pengujian

Pengujian merupakan metode yang dilakukan peneliti untuk mendapatkan data-data yang akan yang digunakan untuk melihat performa dari Progressive Web Apps dan Mobile Web yang dapat diukur berdasarkan fokus pengujian, yaitu performa terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan. Hal ini dilakukan untuk memudahkan dalam memahami bagian serta tahapan yang dilakukan saat pengujian dilakukan. Pengujian sendiri dibagi menjadi dua bagian, yaitu bagian yang membahas pengujian dan bagian yang membahas pengambilan data hasil uji. Pada bagian pengujian dibagi lagi menjadi dua bagian penjabaran variabel pengujian dan skenario pengujian.

3.4.1 Variabel pengujian

Bagian ini akan membahas mengenai variabel-variabel apa saja yang digunakan pada pengujian. Variabel pengujian adalah pengelompokan aspek-aspek yang mempengaruhi performa dari aplikasi-aplikasi yang diuji namun tetap mengarah kepada fokus penelitian. Dengan adanya variabel pengujian akan membantu dalam membuat skenario pengujian pada penelitian ini.

3.4.2 Skenario pengujian

Bagian ini akan membahas mengenai skenario apa saja yang dilakukan saat pengujian dilakukan. Skenario pengujian adalah tahapan-tahapan atau cara-cara terarah dan terstruktur yang dilakukan untuk melakukan pengujian pada

penelitian ini. Skenario pengujian sendiri dibuat berdasarkan variabel-variabel pengujian.

3.5 Pengambilan data hasil uji

Saat pengujian berlangsung, dilakukan pengambilan data-data yang mencerminkan hasil pengujian pada masing-masing skenario pengujian yang dilakukan. Pengambilan data hasil uji adalah penjabaran tentang metode dalam pengambilan data-data saat pengujian berlangsung. Hal ini dilakukan untuk mendefinisikan keterkaitan data-data hasil uji dengan parameter penelitian yang mana membantu dalam menilai dan menganalisis performa Progressive Web Apps dengan Mobile Web terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan nantinya.

Pada saat pengambilan data hasil uji terdapat kemungkinan data yang diambil adalah data yang inkonsisten, dalam arti data tersebut berbeda jauh dengan data yang diambil pada pengujian lainnya walaupun pengujian dilakukan dengan skenario dan kondisi yang sama. Untuk mencegah data hasil uji yang inkonsisten, pengujian dan pengambilan data hasil uji dilakukan sebanyak minimal 10 kali pada tiap-tiap skenario yang ada. Dari data-data yang didapat akan dirata-ratakan untuk dijabarkan pada hasil pengujian.

3.5.1 Data pengujian terkait waktu respon

Waktu Respon adalah total waktu yang dihabiskan untuk berinteraksi dengan peladen dan memuat seluruh berkas gambar, JavaScript, CSS dan berkas eksternal lainnya. Waktu respon sendiri pada dasarnya terdiri dari 6 komponen, yaitu:

1. DNS Lookup Time, adalah waktu yang dibutuhkan untuk mendapatkan IP Address dari sebuah nama domain.
2. TCP Connect, adalah waktu yang dibutuhkan untuk membentuk koneksi ke sebuah halaman *web* setelah DNS Lookup.
3. Redirect Time, adalah waktu yang dibutuhkan untuk mengikuti semua *redirect* untuk menuju ke URL terakhir.
4. First Byte Time, waktu yang dihabiskan untuk menunggu respon dari peladen.
5. Content Time, waktu yang dihabiskan untuk menerima (mengunduh) data respon.
6. DOM Load Time, waktu mulai dari *request* dikirim hingga peramban selesai mengunduh HTML serta selesai membangun Document Object Model (DOM).
7. Page Load Time, waktu halaman dan semua sumber referensi (gambar, CSS, JavaScript dan lainnya) telah dimuat di DOM (SmartBear, 2017).

Dari komponen-komponen tersebut, akan dibentuk menjadi kelompok-kelompok berdasarkan karakteristik proses yang dilakukan. Yang mana, pengambilan data hasil uji akan dilakukan berdasarkan kelompok-kelompok yang dibentuk. Adapun

kelompok yang terbentuk adalah waktu menampilkan halaman dan waktu memuat data.

3.5.1.2 Waktu menampilkan halaman

Waktu menampilkan halaman adalah waktu yang dibutuhkan peramban untuk menampilkan tampilan dari halaman aplikasi. Waktu menampilkan halaman terdiri dari dua komponen, yaitu DOM Load Time dan Page Load Time. Pada Chrome DevTools, proses dari dua komponen ini dibagi menjadi enam tahapan yaitu *loading*, *scripting*, *rendering*, *painting*, *other* dan *idle*. Pengukuran hasil pengujian pada kategori ini berupa total waktu yang dibutuhkan untuk menjalankan keseluruhan tahapan-tahapan tersebut dalam satuan milidetik (mdtk).

3.5.1.3 Waktu memuat data

Waktu memuat data adalah waktu yang dibutuhkan peramban untuk memuat data-data dari jaringan internet ataupun Local Storage yang dibutuhkan oleh halaman aplikasi. Waktu memuat data terdiri dari lima komponen, yaitu DNS Lookup Time, TCP Connect, Redirect Time, First Byte Time dan Content Time. Pada Chrome DevTools, proses dari komponen-komponen ini dibagi menjadi 4 tahapan yaitu *send request*, *receive response*, *receive data*, *parse* HTML. Pengukuran hasil pengujian pada ketegori ini adalah total keseluruhan dari tahapan-tahapan tersebut (*finish loading*) yang juga dalam satuan milidetik (mdtk).

3.5.2 Data pengujian terkait penggunaan memori

Terdapat 2 kategori penggunaan memori, yaitu penggunaan memori per halaman dan penggunaan memori peramban pada sistem operasi. Namun, pada penelitian ini tidak membahas penggunaan memori peramban pada sistem operasi dikarenakan penggunaan memori tersebut sangat kompleks dan sulit untuk dipahami, dan faktanya peluang untuk menafsirkan secara tepat angka-angka data hasil uji yang diambil sangatlah kecil (Kyle, 2016).

Penggunaan memori per halaman adalah besaran memori yang digunakan oleh peramban untuk menjalankan dan memuat beberapa hal, yaitu *code*, *strings*, JS *arrays*, *typed arrays*, *system object*. Pengukuran hasil pengujian pada tahapan ini berupa jumlah keseluruhan memori yang digunakan per halaman dalam satuan *kilobytes* (kB).

3.5.3 Data pengujian terkait penggunaan media penyimpanan

Penggunaan media peyimpanan adalah data-data yang disimpam oleh peramban ataupun Service Worker pada perangkat pengguna. Penggunaan media penyimpanan meliputi Local Storage, *session storage*, indexedDB, Web SQL, *cookies*, *cache storage* dan *application storage*. Pengukuran hasil pengujian pada bagian ini dihitung dari total keseluruhan data aplikasi yang disimpan oleh peramban pada tiap halaman dalam satuan *MegaBytes* (MB).

3.6 Analisis Hasil Pengujian

Bagian ini akan membahas mengenai hasil dan analisis dari hasil pengujian pada penelitian ini. Hal ini dilakukan untuk dapat mengetahui perbandingan performa dari Progressive Web Apps dan Mobile Web terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan. Analisis hasil pengujian dibagi menjadi dua bagian, yaitu bagian yang menjabarkan hasil pengujian dan bagian yang menganalisis dari hasil pengujian pada penelitian ini.

3.6.1 Hasil pengujian

Pada tahap ini adalah penjabaran data-data yang diperoleh dari hasil pengujian yang telah dilakukan. Dari data-data tersebut akan dibandingkan sesuai dengan parameter-parameter penelitian ini, yaitu waktu respon, penggunaan memori dan penggunaan media penyimpanan. Hasil pengujian dan perbandingan dijabarkan dalam bentuk statistika deskriptif, baik berupa tabel ataupun lainnya. Untuk mempermudah dalam penjabaran hasil pengujian, hasil pengujian akan dijabarkan dalam beberapa kategori sesuai dengan skenario pengujian yang dilakukan.

3.6.2 Analisis hasil pengujian

Pada tahap ini akan menganalisis hasil yang telah dijabarkan pada hasil pengujian. Hasil tersebut akan digunakan untuk menganalisis aspek-aspek yang mempengaruhi hasil penelitian untuk mendapatkan kelebihan dan kekurangan dari Mobile Web dan Progressive Web Apps. Analisis hasil pengujian sendiri dibagi menjadi tiga bagian yang mana bagian-bagian tersebut akan menganalisis spesifik terhadap parameter-parameter terkait performa aplikasi. Adapun bagian-bagian tersebut adalah analisis hasil pengujian waktu respon, analisis hasil pengujian penggunaan memori dan analisis hasil pengujian penggunaan media penyimpanan.

3.7 Kesimpulan dan saran

Pengambilan kesimpulan dilakukan oleh penulis dengan mempertimbangkan tahapan-tahapan sebelumnya dan hasil yang diperoleh dari pengujian. Dari kesimpulan ini nantinya akan menjawab rumusan masalah yang sudah dijelaskan. Setelah itu adalah memberikan saran-saran terkait penelitian ini untuk bisa dikembangkan lagi pada penelitian-penelitian selanjutnya.

BAB 4 PEMBAHASAN

Pada bab ini membahas langkah-langkah yang akan dilakukan untuk melakukan pengujian performa pada Progressive Web Apps dan Mobile Web, penggunaan memori dan penggunaan media penyimpanan. Langkah-langkah tersebut meliputi analisis kebutuhan aplikasi, analisis kebutuhan pengujian dan skenario pengujian.

4.1 Analisis Kebutuhan

4.1.1 Perangkat lunak

Pada penelitian ini membutuhkan total empat perangkat lunak pada perangkat komputer dan perangkat bergerak. Pada perangkat komputer membutuhkan 3 perangkat lunak, yaitu XAMPP sebagai peladen, Google Chrome sebagai aplikasi untuk melihat data-data hasil pengujian dan ADB Driver untuk menghubungkan Google Chrome for Android dengan Chrome DevTools. Sedangkan pada perangkat bergerak membutuhkan satu perangkat lunak, yaitu Google Chrome for Android sebagai peramban pada perangkat bergerak. Adapun perangkat lunak yang dibutuhkan pada penelitian ini dapat dilihat pada Tabel 4.1. Sedangkan detail spesifikasi atau versi perangkat lunak yang digunakan saat penelitian ini dapat dilihat pada Tabel 4.2.

Tabel 4.1 Kebutuhan Perangkat Lunak

No	Komputer	Perangkat Bergerak
1	XAMPP: digunakan sebagai peladen yang menyediakan berkas-berkas yang dibutuhkan aplikasi.	Google Chrome for Android: digunakan untuk menjalankan aplikasi yang diujikan.
2	Google Chrome versi 45 keatas: digunakan untuk melihat data-data hasil pengujian yang dilakukan pada Google Chrome for Android berupa respon waktu aplikasi, penggunaan memori halaman *web* dan penggunaan Local Storage oleh aplikasi.	
3	ADB Driver: digunakan untuk menghubungkan Google Chrome for Android dengan Chrome DevTools.	

Tabel 4.2 Perangkat Lunak yang Digunakan

No	Perangkat Lunak	Versi
1	XAMPP	7.0.8 (Windows)
2	Googe Chrome	63.0.3239.84 (Official Build) (64-bit) (Windows)
3	ADB Driver	1.4.3

4.1.2 Perangkat keras

Penelitian ini membutuhkan total tiga perangkat keras, yaitu komputer yang digunakan untuk melihat data-data hasil pengujian, telpon genggam pintar sebagai perangkat bergerak dan kabel USB-C – USB 2.0/3.0 sebagai penghubung antara komputer dengan perangkat bergerak. Adapun spesifikasi minimal dari perangkat keras yang dibutuhkan dalam penelitian ini dapat dilihat pada Tabel 4.3. Sedangkan spesifikasi yang digunakan pada penelitian ini dapat dilihat pada Tabel 4.4.

Tabel 4.3 Kebutuhan Perangkat Keras

No	Perangkat Keras	Spesifikasi Minimal	Jumlah
1	Komputer	Processor Intel Pentium 4 keatas, RAM 1 GB atau lebih, Sistem Operasi Windows 7 keatas	1
2	Telepon Genggam Pintar	Sistem Operasi Android 4.0 keatas	1
3	Kabel USB/USB-C – USB 2.0/3.0 Connector	-	1

Tabel 4.4 Perangkat Keras yang Digunakan

No	Perangkat Keras	Spesifikasi
1	Komputer	Processor Intel Core i5-4200M (2.50GHz 1600MHz 3MB), RAM 4 GB, Sistem Operasi Windows 7
2	Telepon Genggam Pintar	Processor Quad-core (2x1.8 GHz Kryo & 2x1.36 GHz Kryo), RAM 3 GB, Sistem Operasi MIUI V8 (Android 6.0)
3	Kabel USB/USB-C – USB 2.0/3.0 Connector	USB-C – USB 3.0

4.1.3 Lingkungan pengujian

Pada penelitian ini membutuhkan beberapa konfigurasi pada lingkungan pengujian. Konfigurasi tersebut berupa area jaringan, status koneksi jaringan,

kecepatan akses dan opsi pengelolaan *cache*. Area jaringan dibutuhkan untuk mengetahui lingkup jaringan yang digunakan pada saat pengjian. Status koneksi jaringan dibutuhkan untuk menentukan kondisi hubungan peladen dengan perangkat bergerak. Kecepatan akses dibutuhkan untuk menentukan seberapa cepat akses jaringan terhadap proses mengunduh dan mengunggah data dari perangkat ke peladen. Opsi pengelolaan cache dibutuhkan untuk menentukan apakah pengelolaan *cache* dapat dilakukan oleh peramban. Adapun lingkungan pengujian pada penelitian ini adalah sebagai berikut:

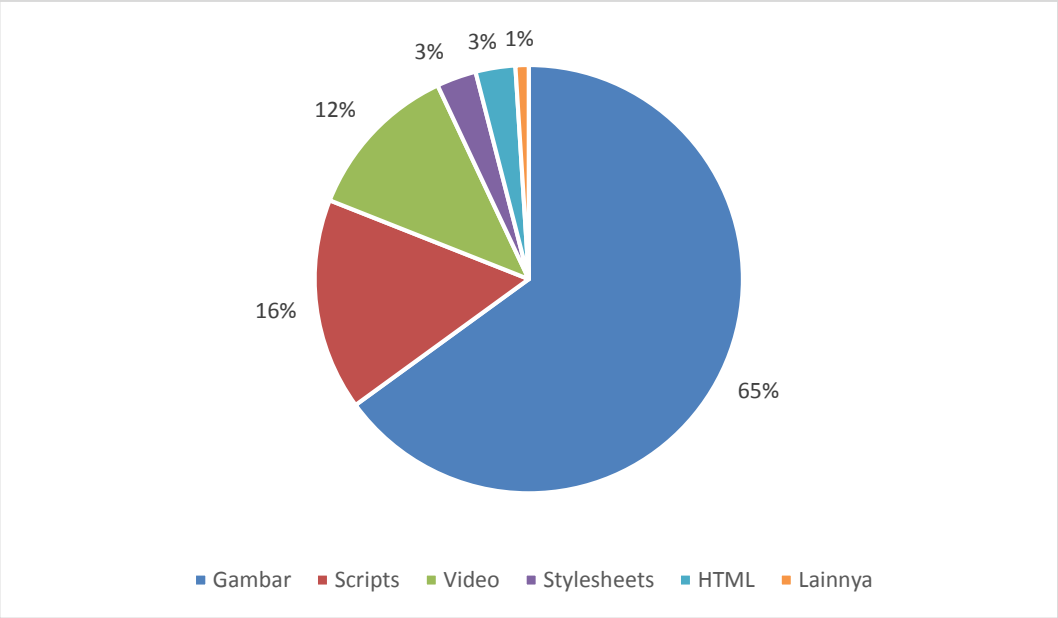
1. Area jaringan: Jaringan Lokal
2. Status koneksi jaringan: terhubung (*online*)
3. Kecepatan akses: 5.9 Megabits per detik (Mbps)
4. Opsi pengelolaan *cache*: Diizinkan (*enabled*)

Area jaringan lokal dipilih agar kecepatan akses jaringan dapat stabil selama proses pengujian. Status koneksi jaringan *online* dipilih agar berkas-berkas dapat diterima oleh perangkat baik pada Progressive Web Apps maupun Mobile Web. Kecepatan akses 5.9 Mbps yang merupakan kecepatan rata-rata akses Internet di Indonesia (Belson, 2016), dikarenakan apabila kecepatan akses adalah kecepatan akses yang disediakan peramban, maka kecepatan akses akan terlalu cepat sehingga akan menyulitkan dalam menganalisis dan melihat perbandingan dari dua tipe aplikasi yang diuji. Opsi pengelolaan cache diizinkan agar Mobile Web dapat mengelola cache seperti pada Progressive Web Apps, akan tetapi pengelolaannya diserahkan pada peramban dan sistem operasi.

4.2 Rancangan Objek Kajian

4.2.1 Struktur Berkas

Aplikasi *web* saat ini rata-rata tersusun dari beberapa tipe berkas dengan proporsinya masing-masing yang dapat dilihat dalam Gambar 4.1 yang tersusun dari sekitar 65% berkas gambar, 16% berkas *scripts*, 12% berkas video, 3% berkas *stylesheets* (CSS), 3% berkas HTML dan berkas lainnya sebanyak kurang dari 1% (Everts, 2015). Aplikasi-aplikasi yang diuji pada penelitian ini dirancang sesuai dengan proporsi tersebut. Berkas gambar yang dimaksud adalah berkas-berkas dengan format JPEG, PNG, SVG atau format lain yang memiliki fungsi menampilkan gambar. Berkas *scripts* yang dimaksud adalah berkas-berkas *client-side scripting* seperti JavaScript. Berkas video adalah berkas-berkas dengan format MP4, 3GP atau format lain yang memiliki fungsi menampilkan video.



Gambar 4.1 Proporsi Berkas Aplikasi Web

Berkas *stylesheets* adalah berkas dengan format CSS atau format lainnya yang memiliki fungsi untuk memberikan corak pada halaman. Berkas HTML adalah berkas dengan format HTML. Berkas lainnya adalah berkas-berkas seperti Web Manifest atau berkas lainnya yang tidak disebutkan diatas. Untuk mempermudah penelitian, keberadaan dari berkas dengan tipe ini menyesuaikan kebutuhan dari masing-masing aplikasi.

Perancangan struktur berkas aplikasi dilakukan pada masing-masing halaman yang ada. Terdapat tiga halaman pada masing-masing aplikasi, yaitu halaman optimal dengan ukuran berkas keseluruhan sebesar kurang lebih 1000 kB, halaman sedang dengan ukuran berkas keseluruhan sebesar kurang lebih 2000 kB dan halaman berat dengan ukuran berkas keseluruhan sebesar kurang lebih 3000 kB. Ukuran-ukuran berkas masing-masing halaman akan dibahas lebih mendetail pada sub bab selanjutnya. Langkah pertama dari perancangan struktur berkas adalah menghitung proporsi dari tipe-tipe berkas yang ada dengan ukuran berkas total masing-masing halaman. Perhitungan proporsi dari tipe-tipe berkas dilakukan dengan cara mengkalikan proporsi yang diacu dari Gambar 3.2 dengan ukuran berkas keseluruhan. Hasil dari perhitungan ini dapat dilihat pada Tabel 4.5.

Tabel 4.5 Proporsi Tipe-tipe Berkas

Tipe / Halaman	Gambar	Scripts	Video	Stylesheets	HTML	Lainnya
Optimal	650 kB	160 kB	120 kB	30 kB	30 kB	10 kB
Sedang	1300 kB	320 kB	240 kB	60 kB	60 kB	20 kB
Berat	1950 kB	480 kB	360 kB	90 kB	90 kB	30 kB

Setelah proporsi dari tipe-tipe berkas dihitung, langkah selanjutnya adalah menyusun halaman aplikasi sesuai dengan proporsinya. Penyusunan berkas dilakukan dengan cara menentukan berkas-berkas apa saja yang dibutuhkan agar suatu tipe berkas sesuai dengan proporsinya masing-masing. Misal, pada tipe berkas gambar pada halaman optimal dibutuhkan enam gambar dengan masing-masing memiliki ukuran berkas 100 kB dan satu gambar dengan ukuran berkas 50 kB yang mana akan menghasilkan total keseluruhan berkas sebesar 650 kB.

Dari proporsi tipe-tipe berkas yang sudah dihitung, ditentukan bahwa halaman optimal tersusun dari beberapa berkas yang dijabarkan pada Tabel 4.6.

Tabel 4.6 Berkas-berkas Halaman Optimal

NO	TIPE	NAMA BERKAS	UKURAN (kB)
1	Gambar	image-small-1.jpg	50
2		image-1.jpg	100
3		image-2.jpg	100
4		image-3.jpg	100
5		image-4.jpg	100
6		image-5.jpg	100
7		image-6.jpg	100
8	Scripts	script.js	10
9		jquery-3.2.1.min.js	85
10		responsiveTabs.js	8
11		owl.carousel.min.js	42
12		modernizr.min.js	11
13		featherlight.min.js	9
14	Video	video-120.webm	122
15	Stylesheets	style.css	11
16		owl.carousel.css	4
17		owl.theme.default.css	2
18		responsive-tabs.css	3
19		featherlight.css	4
20		modernizr.css	7
21		modernizr.reset.css	2
22	HTML	index.html	30
TOTAL			**1000**

Pada halaman ini juga dibutuhkan beberapa *library* yang berfungsi untuk memudahkan dalam implementasi aplikasi, seperti *library* Owlcarousel yang digunakan untuk membuat susunan gambar yang dapat digeser, Modernizer yang digunakan untuk membuat tab pada halaman, Featherlight yang digunakan untuk membuat *popup* pada halaman dan lain sebagainya. Selain berfungsi untuk memudahkan dalam implementasi aplikasi, penggunaan *library* juga berfungsi

untuk menambahkan ukuran berkas agar sesuai dengan proporsi dari masing-masing tipe berkas. Semua berkas-berkas pada halaman ini akan disisipkan atau dipanggil pada berkas index.html.

Struktur berkas pada halaman sedang dan halaman berat tidak jauh berbeda dengan struktur berkas pada halaman optimal, yang membedakan dari masing-masing halaman adalah jumlah *library* yang digunakan serta banyaknya objek yang dimiliki oleh halaman tersebut. Hal ini dilakukan karena tiap-tiap halaman hanya dibedakan dari total ukuran berkas total yang dibutuhkan per halaman. Adapun struktur berkas pada halaman sedang dapat dilihat pada Tabel 4.7 dan struktur berkas pada halaman berat dapat dilihat pada Tabel 4.8.

Tabel 4.7 Berkas-berkas halaman sedang

NO	TIPE	NAMA	UKURAN (kB)
1	Gambar	image-small-1.jpg	50
2		image-1.jpg	100
3		image-2.jpg	100
4		image-3.jpg	100
5		image-4.jpg	100
6		image-5.jpg	100
7		image-medium-1.jpg	175
8		image-medium-2.jpg	175
9		image-large-1.jpg	200
10		image-large-2.jpg	200
11	*Scripts*	script.js	10
12		jquery-3.2.1.js	262
13		owl.carousel.min.js	42
14		modernizr.min.js	11
15		featherlight.min.js	9
16	Video	video-240.webm	240
17	*Stylesheet*	style.css	11
18		owl.carousel.css	4
19		owl.theme.default.css	2
20		featherlight.css	4
21		modernizr.css	7
22		modernizr.reset.css	2
23		jquery.ui.css	32
24	HTML	index.html	61
TOTAL			**1997**

Tabel 4.8 Berkas-berkas halaman berat

NO	TIPE	NAMA	UKURAN (kB)
1	Gambar	image-small-1.jpg	50

2		image-1.jpg	100
3		image-2.jpg	100
4		image-3.jpg	100
5		image-4.jpg	100
6		image-5.jpg	100
7		image-6.jpg	100
8		image-medium-1.jpg	175
9		image-medium-2.jpg	175
10		image-medium-3.jpg	175
11		image-medium-4.jpg	175
12		image-large-1.jpg	200
13		image-large-2.jpg	200
14		image-large-3.jpg	200
15	*Scripts*	script.js	10
16		jquery-3.2.1.js	85
17		owl.carousel.js	84
18		modernizr.js	51
19		featherlight.min.js	9
20		jquery-ui.min.js	248
21	Video	video-360.webm	360
22	*Stylesheet*	style.css	11
23		owl.carousel.css	4
24		owl.theme.default.css	2
25		featherlight.css	4
26		modernizr.css	7
27		modernizr.reset.css	2
28		jquery.ui.css	32
29		jquery-ui.structure.min.css	16
30		jquery-ui.theme.min.css	14
31	HTML	index.html	90
TOTAL			**2979**

Terdapat sedikit perbedaan dari halaman-halaman pada Progressive Web Apps. Halaman pada Progressive Web Apps membutuhkan berkas Web App Manifest serta berkas JavaScript yang mengatur manajemen *cache* (Service Worker). Pada Web App Manifest, dicantumkan beberapa informasi dari aplikasi. Informasi-informasi tersebut berupa nama lengkap aplikasi (name), nama singkat aplikasi (short_name), gambar ikon (icons), URL awal (start_url), tampilan (display), warna latar (background_color) dan warna tema (theme_color) yang ditulis dalam format JSON. Adapun isi dari Web App Manifest dijabarkan dalam Kode Program 4.1.

```
1 {
```

```
  "name": "Pengujian Progressive Web Apps",
  "short_name": "Pengujian PWA",
  "icons": [{
      "src": "images/favicon.png",
      "sizes": "144x144",
      "type": "image/png"
    }],
  "start_url": "/index.html",
  "display": "standalone",
  "background_color": "#FFFFFF",
  "theme_color": "#1E6BEC"
}
```

Kode Program 4.1 Web App Manifest (manifest.json)

Pengaturan manajemen *cache* diatur pada berkas service-worker.js yang akan dibahas lebih mendetail pada sub bab selanjutnya. Dengan bertambahnya jumlah berkas yang dibutuhkan pada Progressive Web Apps, maka ukuran berkas dari masing-masing halaman pada Progressive Web Apps juga bertambah. Total ukuran berkas halaman pada Progressive Web Apps bertambah 14 kB, yakti 1 kB dari berkas `manifest.json`, 1 kB dari berkas app.js dan 12 kB dari berkas service-worker.js.

4.2.2 Antarmuka

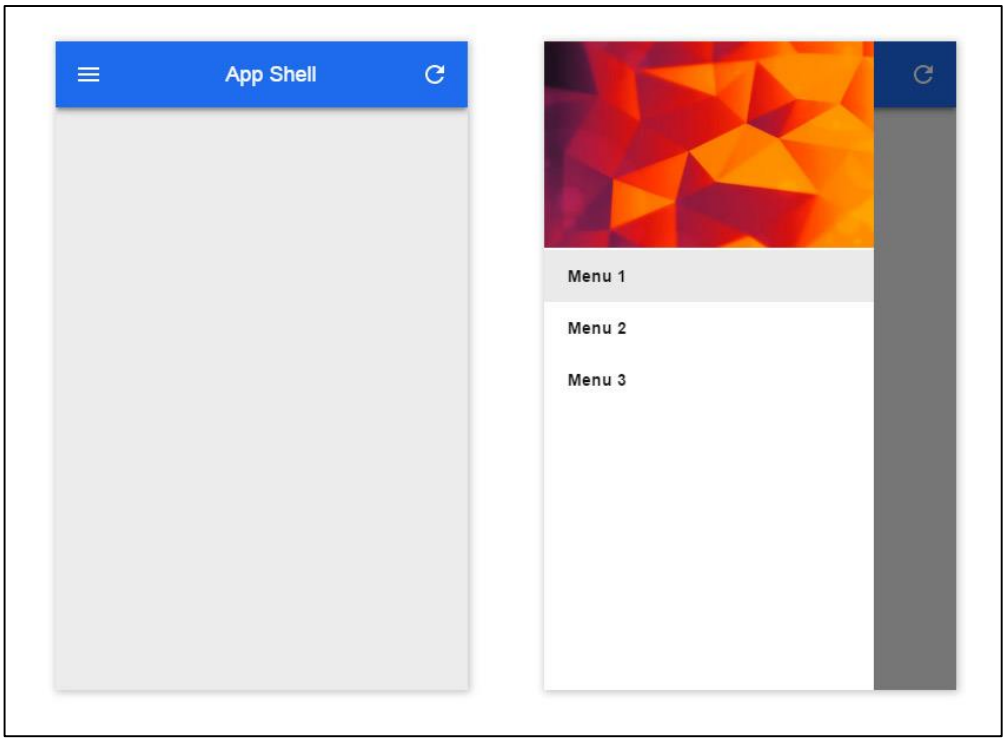
Antarmuka halaman aplikasi dibagi menjadi dua bagian, yaitu kerangka halaman dan konten halaman. Bentuk dari kerangka halaman sendiri mengikuti pedoman Application Shells pada Progressive Web Apps. Bagian kerangka halaman pada aplikasi-aplikasi yang diuji dibagi menjadi tiga blok, yaitu blok header yang menampilkan header halaman, blok sidebar yang menampilkan menu sisi halaman serta blok konten yang digunakan untuk membungkus bagian konten halaman. Tampilan kerangka dapat dilihat dalam Gambar 4.2. Konten halaman tersusun dari beberapa blok. Terdapat tiga jenis blok pada bagian konten halaman, yaitu blok tulisan, blok gambar dan blok video dengan kegunaannya masing-masing yang dapat dilihat pada Tabel 4.9.

Tabel 4.9 Blok Penyusun Antarmuka

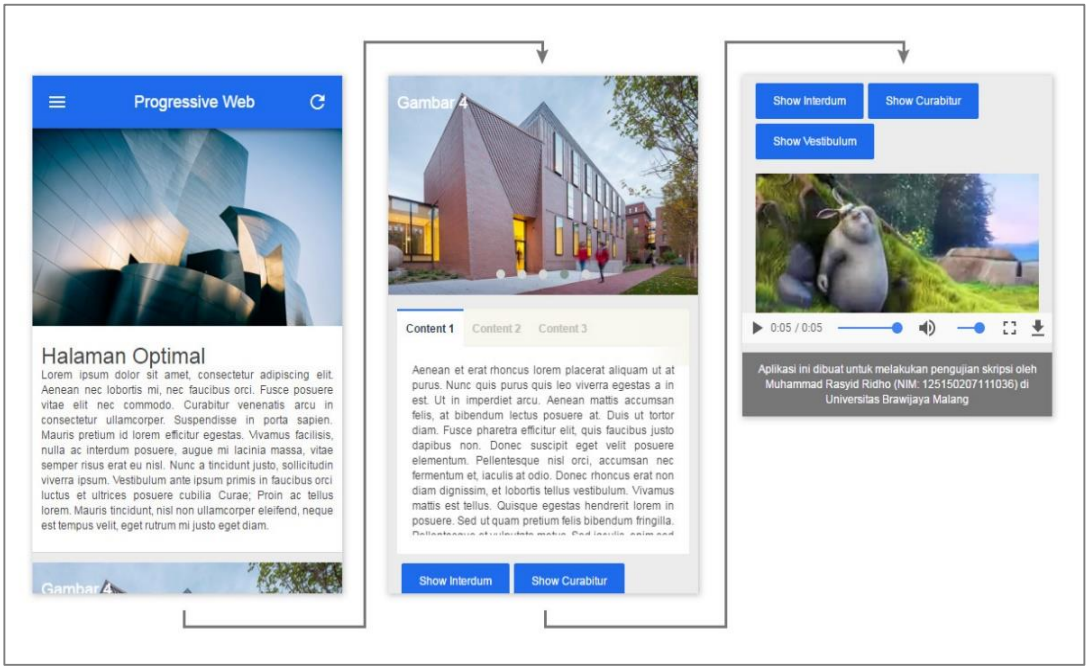
No	Nama Blok	Kegunaan
1	Teks	Menampilkan teks pada halaman
2	Gambar	Menampilkan satu gambar pada halaman
3	Video	Menampilkan satu video pada halaman

Isi atau konten masing-masing blok menyesuaikan dengan struktur berkas pada tiap-tiap halaman. Contohnya struktur berkas gambar pada halaman optimal yang terdiri dari tujuh berkas gambar, maka terdapat tujuh blok gambar pada halaman tersebut. Urutan peletakan blok pada antarmuka halaman tidak memiliki aturan khusus, contohnya Blok Teks bisa diletakkan di awal halaman, di tengah, di akhir, setelah Blok Gambar dan lain sebagainya. Begitupun dengan blok-blok yang lain karena yang diujikan adalah besaran berkas bukan kualitas konten dari suatu halaman. Adapun antarmuka halaman optimal dapat dilihat dalam Gambar 4.3,

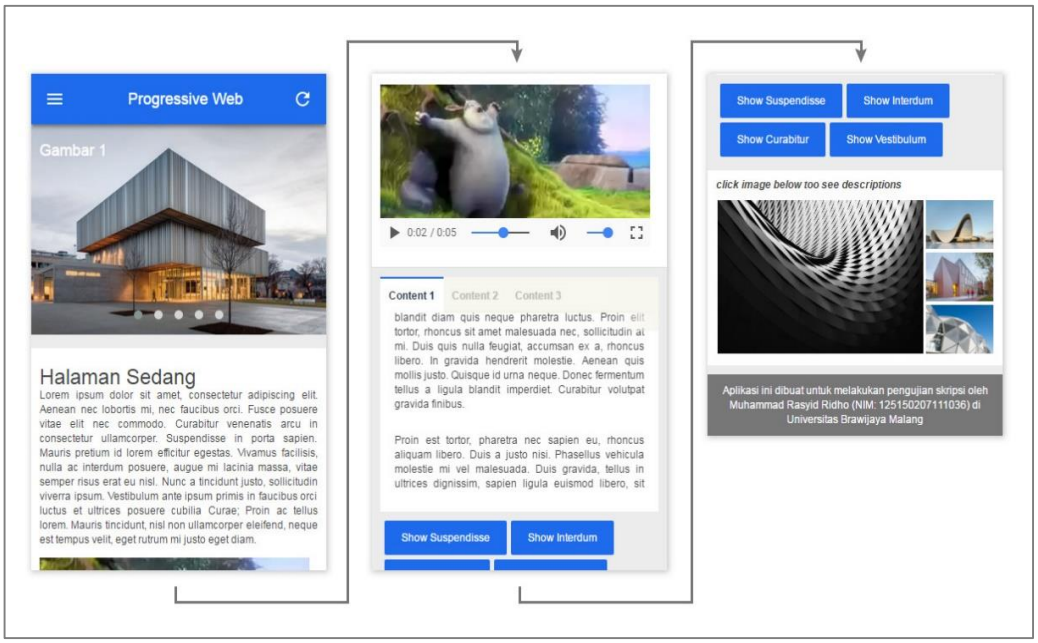
sedangkan antarmuka halaman sedang dapat dilihat dalam Gambar 4.4 dan antarmuka halaman berat dapat dilihat dalam Gambar 4.5.



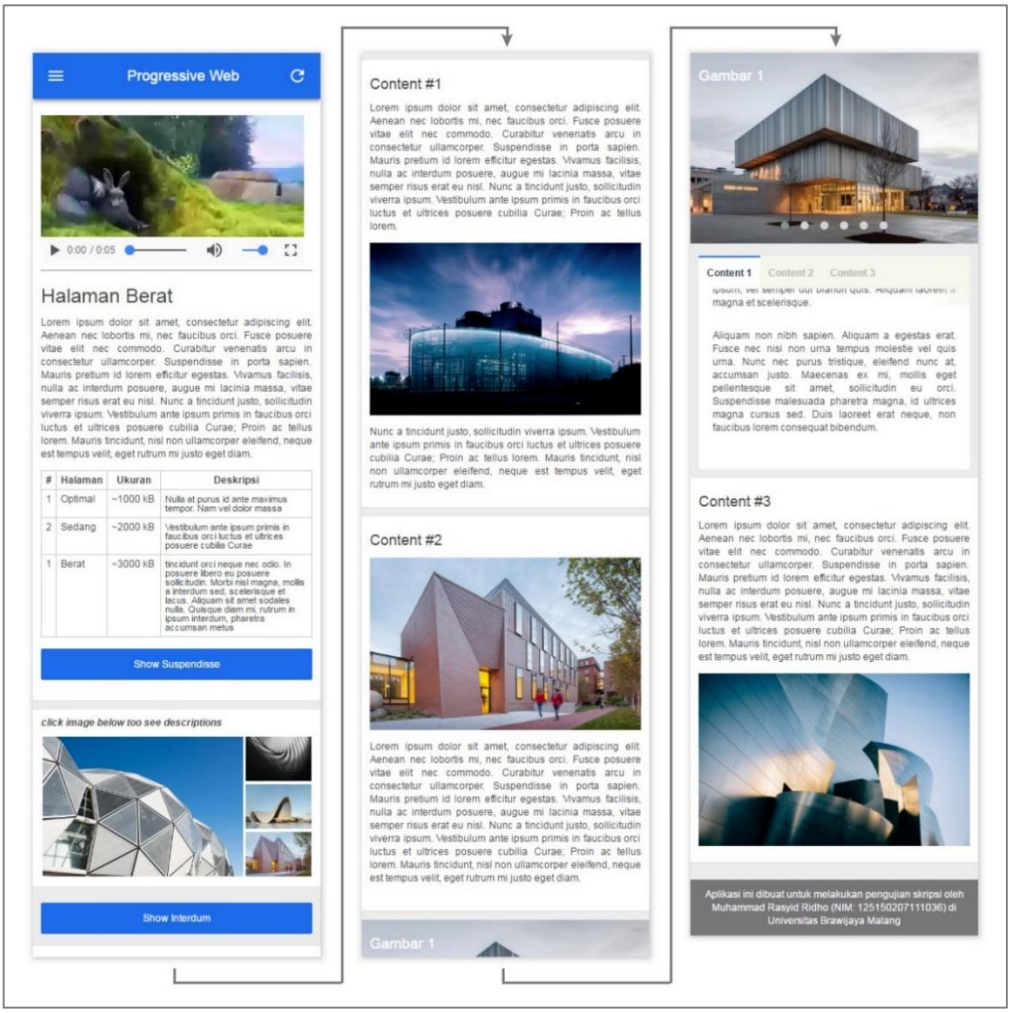
Gambar 4.2 Tampilan Kerangka Halaman



Gambar 4.3 Antarmuka halaman optimal



Gambar 4.4 Antarmuka halaman sedang



Gambar 4.5 Antarmuka halaman berat

4.3 Pengujian

4.3.1 Variabel pengujian

Terdapat empat variabel pengujian pada penelitian ini, yaitu tipe aplikasi, fungsionalitas aplikasi, pengaksesan halaman aplikasi dan ukuran *cache*. Variabel-variabel ini akan digunakan sebagai acuan dalam membuat skenario pengujian, dalam arti skenario pengujian yang dibuat terdiri dari kombinasi-kombinasi dari nilai-nilai yang ada pada tiap-tiap variabel yang ada.

4.3.1.1 Tipe aplikasi

Terdapat dua tipe aplikasi yang diujikan pada penelitian ini, yaitu Mobile Web dan Progressive Web Apps. Tiap tipe aplikasi memiliki ketentuan dan batasan tersendiri yang ditentukan dan dibuat serupa mungkin dengan tipe aplikasi yang lain, serupa dalam artian memiliki fungsional, tampilan, alur kerja dan mengakses ke *server* yang sama. Namun ukuran berkas dari tiap aplikasi dapat berbeda, karena teknologi yang digunakan pada dua tipe aplikasi ini berbeda. Contoh saja pada Progressive Web Apps membutuhkan berkas tambahan seperti berkas Service Worker, manifest dan lain sebagainya. Teknologi-teknologi yang digunakan pada masing-masing tipe aplikasi dapat dilihat pada Tabel 4.10. Untuk memudahkan dalam pengembangan aplikasi, tiap tipe aplikasi juga menggunakan *library* JavaScript, yaitu jQuery. Sedangkan struktur HTML dan berkas CSS akan dibuat sama pada masing-masing tipe aplikasi. Pada Mobile Web, tidak dibuat *script* khusus dalam manajemen *cache*, dalam artian *cache* akan murni diatur oleh peramban dan sistem operasi.

Tabel 4.10 Tipe aplikasi

No	Tipe Aplikasi	Teknologi yang digunakan
1	Mobile Web	HTML, CSS, JavaScript
2	Progressive Web Apps	HTML, CSS, JavaScript, Service Worker, Web App Manifest, app shell

Tiap-tiap tipe aplikasi tersusun dari berkas-berkas dengan format tersendiri seperti HTML, CSS, JavaScript, gambar dan video pada Mobile Web dan ditambah Wep App Manifest pada Progressive Web Apps. Terdapat dua tipe berkas pada penelitian ini, yaitu berkas global yaitu berkas yang digunakan bersama-sama oleh tipe-tipe aplikasi dan berkas spesifik yang hanya digunakan oleh tipe aplikasi tertentu. Berkas global terdiri dari berkas-berkas gambar, video, *font*, *library* JavaScript dan berkas pada sisi *server* (*server-side scripting*). Berkas-berkas gambar, video dan *font* digunakan untuk membentuk antar muka aplikasi. *Library* JavaScript digunakan untuk membantu dalam pengembangan aplikasi baik dari sisi antar muka aplikasi, jembatan antara antar muka dengan *back-end* aplikasi serta implementasi fitur Progressive Web Apps (Service Worker).

4.3.1.2 Fungsional aplikasi

Fungsional aplikasi adalah fitur atau fungsi yang dimiliki oleh aplikasi, dalam hal ini adalah fitur atau fungsi aplikasi dalam menampilkan konten dari halaman-halaman dengan ukuran berkas dan metode pemanggilan data yang berbeda. Ukuran berkas yang dimaksud adalah ukuran berkas dari keseluruhan kebutuhan dari suatu halaman. Ukuran berkas dibagi menjadi tiga kategori, yaitu ukuran yang optimal, sedang dan berat. Besar ukuran halaman yang optimal adalah ukuran halaman yang tidak lebih dari 1 *MegaBytes* (MB) atau 1000 *kiloBytes* (kB), sedangkan ukuran halaman yang berat adalah ukuran halaman yang diproyeksikan akan dibutuhkan oleh halaman-halaman *web* pada tahun 2017, yaitu 3 MB (Everts, 2015). Ukuran halaman yang sedang adalah nilai tengah dari halaman yang optimal dan halaman yang berat, yang juga merupakan rata-rata ukuran halaman pada tahun 2016, yaitu sekitar 2 MB (Cremin, 2016). Adapun deskripsi dari masing-masing fungsional aplikasi dapat dilihat pada Tabel 4.11 dibawah ini.

Tabel 4.11 Kebutuhan Fungsionalitas Aplikasi

No	Nama Fungsi	Deskripsi
1	Halaman optimal	Menampilkan halaman yang berupa konten teks dan gambar dengan ukuran berkas keseluruhan tidak lebih dari atau sekitar 1000 kB
2	Halaman sedang	Menampilkan halaman yang berupa konten teks dan gambar dengan ukuran berkas keseluruhan sekitar 2000 kB
3	Halaman berat	Menampilkan halaman yang berupa konten teks dan gambar dengan ukuran berkas keseluruhan sekitar 3000 kB

4.3.1.3 Pengaksesan halaman aplikasi

Pengaksesan halaman aplikasi adalah kondisi pengaksesan suatu halaman, yaitu kondisi ketika suatu halaman aplikasi belum pernah diakses dan kondisi ketika suatu halaman aplikasi sudah pernah diakses sebelumnya dari suatu perangkat bergerak. Hal ini digunakan untuk mengetahui perbedaan performa aplikasi ketika berkas-berkas aplikasi belum tersedia dan ketika berkas-berkas aplikasi sudah tersimpan pada Local Storage perangkat bergerak, baik berupa *cache*, *session*, *cookies* dan lain sebagainya. Adapun deskripsi dari kondisi-kondisi pengaksesan yang dijadikan sebagai variabel pengujian pada penelitian ini dijabarkan pada Tabel 4.12.

Tabel 4.12 Pengaksesan halaman aplikasi

No	Urutan pengaksesan	Deskripsi
1	Pengaksesan ke-1	Pada kondisi ini aplikasi belum memiliki berkas-berkas yang tersimpan pada Local Storage.

2	Pengaksesan ke-2	Pada kondisi ini aplikasi sudah menyimpan berkas-berkas yang dibutuhkan pada Local Storage.

4.3.1.4 Ukuran berkas cache

Ukuran berkas *cache* yang di maksud adalah jumlah ukuran berkas-berkas yang disimpan oleh aplikasi pada Local Storage. Hal ini perlu dimasukkan untuk mengetahui bagaimana dampak yang diberikan oleh Service Worker terhadap performa aplikasi (Progressive Web Apps) dibandingkan dengan aplikasi yang tidak menggunakan Service Worker. Karena jumlah *cache* pada suatu aplikasi dinamis terhadap kebutuhan dalam arti bisa 0% - 100% dari berkas aplikasi, maka dibutuhkan beberapa kali pengujian dengan persentase *cache* yang berbeda dengan maksud untuk mengetahui seberapa besar efisiensi penggunaan Service Worker terhadap performa aplikasi berdasarkan persentase ukuran *cache*. Pengujian dengan variabel ini dilakukan pada fungsional halaman optimal pada aplikasi Progressive Web Apps saja dikarenakan pada Mobile Web tradisional tidak memiliki teknologi dalam mengatur *cache* layaknya pada Progressive Web Apps. Untuk memudahkan pengujian pada penelitian ini, persentase menggunakan angka dengan inkremental 10 yang dimulai dari 0 hingga 100, lebih jelasnya dijabarkan pada Tabel 4.13. Pemilihan berkas-berkas *cache* dilakukan dengan cara mengestimasi jumlah ukuran beberapa berkas yang sudah dijabarkan pada Tabel 4.6, yang mana estimasi jumlah tersebut dapat mendekati ukuran *cache* keseluruhan pada masing-masing persentase yang diberikan.

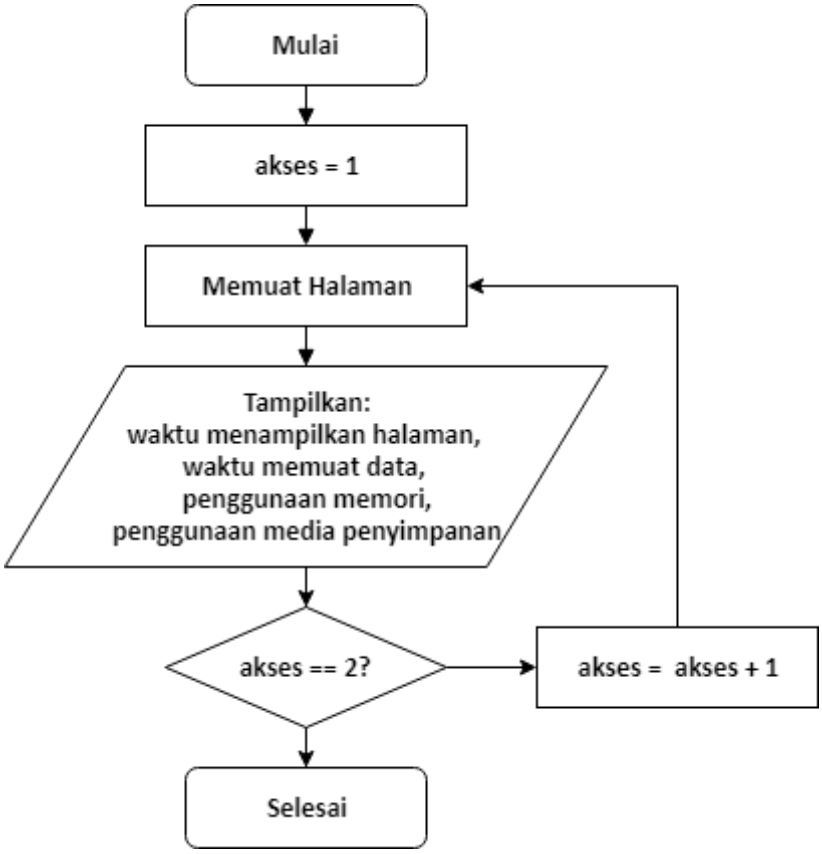
Tabel 4.13 Ukuran Berkas *Cache*

No	Persentase	Ukuran *Cache* Keseluruhan	Berkas-berkas *Cache*
1	0%	0 kB	-
2	10%	~ 100 kB	index.html, style.css, modernizr.css, script.js, owl.carousel.min.js
3	20%	~ 200 kB	index.html, style.css, modernizr.css, script.js, owl.carousel.min.js, image-1.jpg.
4	30%	~ 300 kB	index.html, style.css, modernizr.css, script.js, owl.carousel.min.js, image-1.jpg, image-2.jpg
5	40%	~ 400 kB	index.html, style.css, modernizr.css, script.js, owl.carousel.min.js, image-1.jpg, image-2.jpg, image-3.jpg
6	50%	~ 500 kB	index.html, style.css, modernizr.css, script.js, owl.carousel.min.js, image-1.jpg, image-2.jpg, image-3.jpg, image-4.jpg
7	60%	~ 600 kB	60%: index.html, style.css, modernizr.css, script.js, owl.carousel.min.js, image-1.jpg, image-2.jpg, image-3.jpg, image-4.jpg, image-5.jpg

8	70%	~ 700 kB	70%: index.html, style.css, modernizr.css, script.js, owl.carousel.min.js, image-1.jpg, image-2.jpg, image-3.jpg, image-4.jpg, image-5.jpg, image-6.jpg
9	80%	~ 800 kB	index.html, style.css, modernizr.css, owl.carousel.css, owl.theme.default.css, responsive-tabs.css, featherlight.css, modernizr.reset.css, script.js, jquery-3.2.1.min.js, owl.carousel.min.js, image-1.jpg, image-2.jpg, image-3.jpg, image-4.jpg, image-5.jpg, image-6.jpg
10	90%	~ 900 kB	index.html, style.css, script.js, jquery-3.2.1.min.js, owl.carousel.min.js, image-1.jpg, image-2.jpg, image-3.jpg, image-4.jpg, image-5.jpg, image-6.jpg, video-120.webm
11	100%	~ 1000 kB	Seluruh berkas

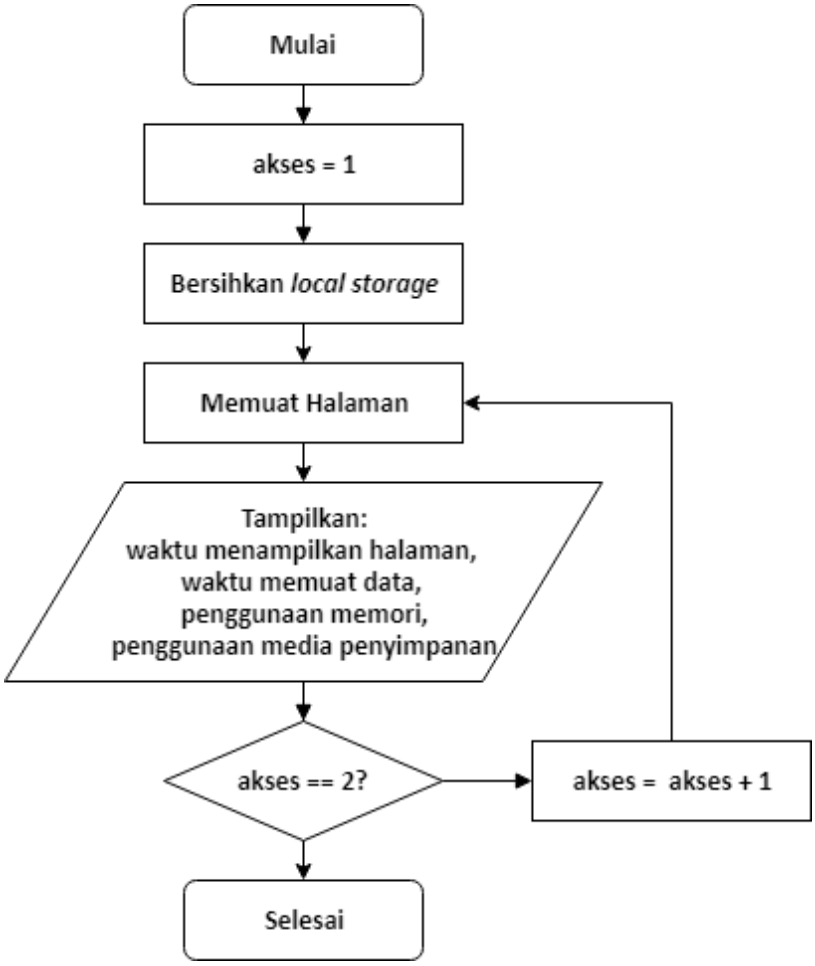
4.3.2 Skenario pengujian

Dalam penelitian ini terdapat dua skenario, yaitu:



Gambar 4.6 Diagram Alir Skenario Pengujian 1

1. Skenario 1 dengan variabel pengujian berupa tipe aplikasi, pengaksesan halaman aplikasi dan fungsional aplikasi (halaman optimal, halaman sedang, halaman berat). Adapun diagram alir dari tahapan-tahapan yang ada pada skenario 1 dijabarkan dalam Gambar 4.6. Skenario ini dimulai dengan pengaksesan ke-1 untuk memuat halaman pada tiap-tiap aplikasi dan fungsional aplikasi. Setelah halaman selesai memuat data pengujian yaitu waktu menampilkan halaman, waktu memuat data, penggunaan memori dan penggunaan media penyimpanan. Data pengujian tersebut diambil dari Chrome DevTools. Lalu dilakukan hal serupa kembali untuk pengaksesan ke-2.
2. Skenario 2 dengan variabel pengujian berupa tipe aplikasi (Progressive Web Apps), pengaksesan halaman aplikasi dan fungsional aplikasi (halaman berat). Adapun diagram alir dari tahapan-tahapan yang ada pada skenario 2 dijabarkan dalam Gambar 4.7.



Gambar 4.7 Diagram Alir Skenario Pengujian 2

Skenario ini dimulai dengan membersihkan Local Storage dari *cache* sebelumnya, lalu melakukan pengaksesan ke-1 untuk memuat halaman pada tiap-tiap aplikasi dan fungsional aplikasi. Setelah halaman selesai memuat data pengujian yaitu waktu menampilkan halaman, waktu memuat data pengujian, penggunaan memori dan penggunaan media penyimpanan. Data pengujian

tersebut diambil dari Chrome DevTools. Lalu dilakukan hal serupa kembali untuk pengaksesan ke-2 tetapi tanpa membersihkan Local Storage terlebih dahulu.

BAB 5 ANALISIS

5.1 Hasil Pengujian

Setelah pengujian dilakukan, data-data hasil pengujian dijabarkan berdasarkan skenario pengujian yang dilakukan. Terdapat dua satuan nilai pada hasil pengujian penelitian ini, yaitu satuan waktu yang dijabarkan dalam milidetik (mdtk) dan satuan ukuran berkas yang dijabarkan dalam *kiloBytes* (kB). Hasil pengujian akan dijabarkan dalam bentuk tabel-tabel dimana didalam masing-masing tabel tedapat kolom-kolom yang mewakilkan variabel uji dan hasil dari pengujian terhadap variabel tersebut.

Hasil pengujian dibagi menjadi tiga kelompok sesuai dengan parameter pada penelitian ini. Kelompok pertama adalah penjabaran parameter waktu respon yang diwakilkan oleh dua kolom, yaitu kolom yang berisi waktu untuk menampilkan halaman (Waktu Tampil Halaman) dan kolom yang berisi waktu untuk memuat data (Waktu Muat Data). Kelompok kedua adalah penjabaran penggunaan memori yang diwakilkan oleh tiga kolom, yaitu kolom yang berisi jumlah penggunaan memori oleh berkas halaman (Memori Halaman), kolom yang berisi jumlah penggunaan memori oleh berkas Service Worker (Memori SW) dan kolom total penggunaan halaman keseluruhan (Memori Total) yang didapat dari hasil penjumlahan kolom Memori Halaman dan Memori SW. Namun, khusus pada tipe aplikasi yang terkait dengan Mobile Web, kolom Memori Halaman dan Memori SW tidak dijabarkan dikarenakan penggunaan memori hanya berasal dari memori oleh berkas halaman saja. Kelompok ketiga adalah penjabaran penggunaan media penyimpanan yang diwakilkan oleh satu kolom saja. Kolom yang dimaksud adalah kolom yang berisi jumlah penggunaan media penyimpanan pada tiap halaman (Media Penyimpanan).

5.1.1 Hasil pengujian skenario 1

Hasil pengujian skenario 1 dibagi menjadi 2 tabel, yaitu pengujian pada Mobile Web yang dijabarkan pada Tabel 5.1 dan pengujian pada Progressive Web Apps yang dijabarkan pada Tabel 5.2. Pada masing-masing tabel terdapat dua kolom yang mengelompokkan variabel pengujian. Variabel pertama adalah fungsional aplikasi (Halaman) yang terdiri dari halaman optimal, sedang dan berat. Sedangkan variabel kedua adalah urutan pengaksesan halaman aplikasi (Akses Ke) yang terdiri dari pengaksesan halaman ke-1 dan ke-2.

Tabel 5.1 Hasil Pengujian Skenario 1 – Mobile Web

Halaman	Akses Ke	Waktu Tampil Halaman (mdtk)	Waktu Muat Data (mdtk)	Total Memori (kB)	Media Penyimpanan (kB)
Optimal	1	1761	1254	5146	0
	2	1507	1200	5143	0

Sedang	1	2898	2673	5230	0
	2	2662	2456	5235	0
Berat	1	4305	3864	6167	0
	2	3779	3707	6170	0

Tabel 5.2 Hasil Pengujian Skenario 1 – Progressive Web Apps

Halaman	Akses Ke	Waktu Tampil Halaman (mdtk)	Waktu Muat Data (mdtk)	Memori Halaman (kB)	Memori SW (kB)	Total Memori (kB)	Media Penyimpanan (kB)
Optimal	1	1898	1613	5170	2350	7520	67
	2	1879	1844	5196	2391	7587	68
Sedang	1	2971	2763	5222	2350	7572	94
	2	2568	2938	5276	2392	7668	95
Berat	1	4374	3856	6219	2350	8569	124
	2	3506	4393	6233	2392	8625	126

5.1.2 Hasil pengujian skenario 2

Meski pada skenario 2 hanya difokuskan pada variabel ukuran berkas *cache* pada halaman optimal Progressive Web Apps, demi membantu dalam membandingkan dua tipe aplikasi dan mempermudah memahami analisis penelitian nanti, ditambahkan juga hasil pengujian halaman optimal dari Mobile Web secara umum. Oleh karena itu hasil pengujian skenario 2 dibagi menjadi dua tabel, yaitu Tabel 5.3 yang menjabarkan hasil pengujian halaman optimal dari Mobile Web secara umum dan Tabel 5.4 yang menjabarkan hasil pengujian halaman optimal dari Progressive Web Apps secara lebih mendetail.

Pada Tabel 5.4 terdapat dua kolom yang mengelompokkan variabel pengujian. Variabel pertama adalah ukuran berkas *cache* yang diujikan (*Cache*) dan variabel kedua adalah urutan pengaksesan halaman aplikasi (Akses Ke) yang terdiri dari pengaksesan halaman ke-1 dan ke-2. Pada Tabel 5.4 tidak dijabarkan variabel ukuran berkas *cache* dikarenakan ukuran *cache* pada Mobile Web bersifat statis, dalam arti diserahkan total pada aplikasi peramban.

Tabel 5.3 Hasil Pengujian Skenario 2 – Mobile Web

Akses Ke	Waktu Tampil Halaman (mdtk)	Waktu Muat Data (mdtk)	Total Memori (kB)	Media Penyimpanan (kB)
1	1761	1254	5146	0
2	1507	1200	5143	0

Tabel 5.4 Hasil Pengujian Skenario 2 – Progressive Web Apps

Cache	Akses Ke	Waktu Tampil Halaman (mdtk)	Waktu Muat Data (mdtk)	Memori Halaman (kB)	Memori SW (kB)	Total Memori (kB)	Media Penyimpanan (kB)
0%	1	1915	1285	5212	2304	7516	11
	2	1966	1520	5183	2356	7539	11
10%	1	1912	1284	5199	2350	7549	115
	2	1889	1578	5195	2393	7588	145
20%	1	1934	1268	5195	2351	7546	216
	2	1852	1588	5192	2394	7586	246
30%	1	1926	1282	5193	2351	7544	317
	2	1790	1561	5193	2395	7588	347
40%	1	1961	1287	5194	2351	7545	418
	2	1748	1532	5197	2396	7593	448
50%	1	1962	1287	5197	2350	7547	519
	2	1687	1553	5200	2397	7597	549
60%	1	1963	1293	5198	2350	7548	621
	2	1629	1552	5200	2398	7598	651
70%	1	1987	1226	5202	2350	7552	722
	2	1330	1581	5200	2399	7599	752
80%	1	1999	1286	5202	2350	7552	823
	2	1296	1527	5199	2404	7603	853
90%	1	1970	1239	5198	2350	7548	924
	2	1246	1946	5197	2399	7596	954
100%	1	1990	1267	5202	2351	7553	1026
	2	1177	1939	5204	2408	7612	1056

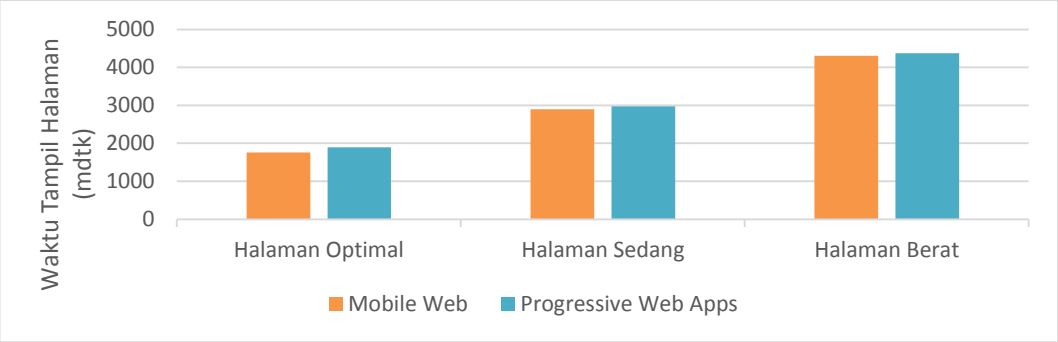
5.2 Analisis Hasil Pengujian

Membahas skenario-skenario yang akan digunakan untuk menguji Progressive Web Apps dan Mobile Web terkait waktu respon, penggunaan memori dan penggunaan media penyimpanan sesuai dengan analisis kebutuhan pengujian.

5.2.1 Analisis hasil pengujian waktu respon

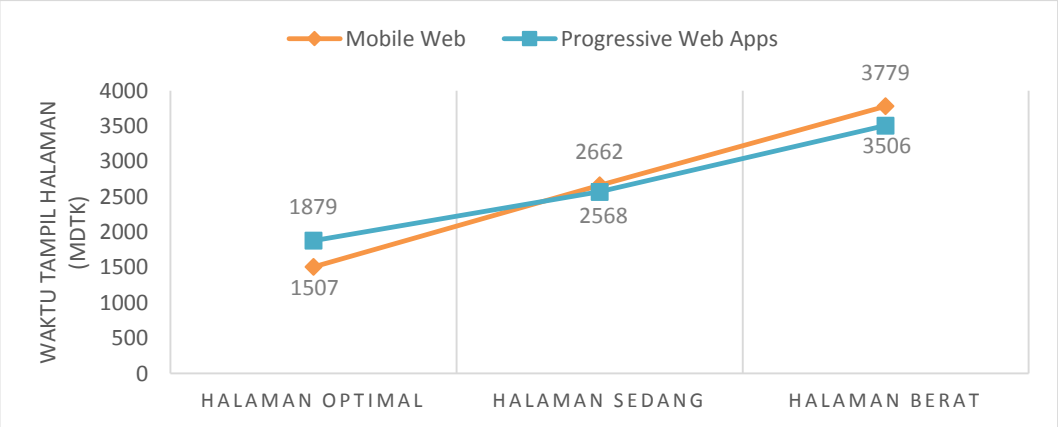
Pada hasil pengujian skenario 1, dua kolom yang merepresentasikan hasil performa dari waktu respon yaitu Waktu Tampil Halaman dan Waktu Muat Data. Untuk kolom Waktu Tampil Halaman, hasil pengujian menunjukkan bahwa pada pengaksesan pertama dengan ukuran berkas halaman optimal, sedang maupun berat, Mobile Web masih lebih cepat dari Progressive Web Apps. Hal ini dikarenakan ada beberapa proses-proses tambahan yang dilakukan oleh Progressive Web Apps dan tidak pada Mobile Web. Contohnya pendaftaran Service Worker, penyimpanan Application Shell pada *cache* dan lain sebagainya. Walaupun demikian, tidak ada selisih perbedaan yang signifikan antara Mobile

Web dan Progressive Web Apps. Adapun grafik yang menggambarkan hal ini dapat dilihat dalam Gambar 5.1.



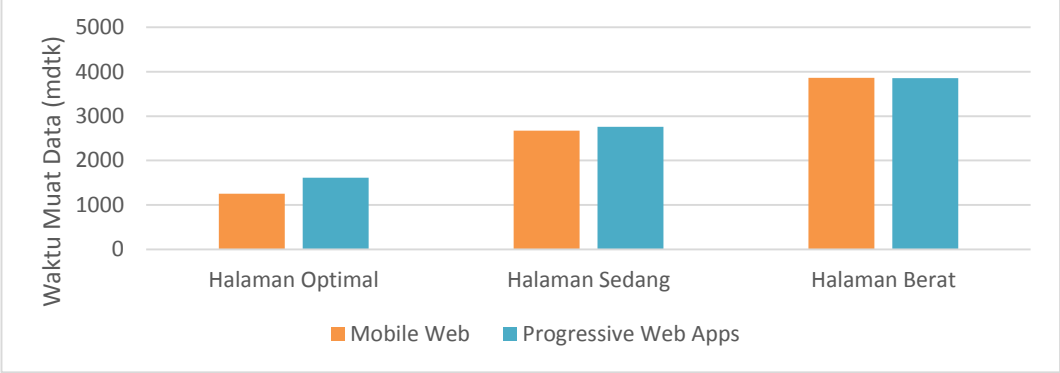
Gambar 5.1 Perbandingan Waktu Tampil Halaman Pengaksesan 1 Skenario 1

Pada pengaksesan ke-2, Mobile Web kembali unggul pada halaman optimal. Namun, tidak pada halaman sedang dan halaman berat, dimana Progressive Web Apps menunjukkan waktu yang lebih cepat. Pada halaman optimal Mobile Web sekitar 300 mdtk lebih cepat dari Progressive Web Apps. Sedangkan pada halaman sedang, Progressive Web Apps sekitar 100 mdtk lebih cepat dari Mobile Web; pada halaman berat, Progressive Web Apps kembali unggul dengan waktu sekitar 200 mdtk lebih cepat dari Mobile Web. Melihat selisih kecepatan antara dua tipe aplikasi ini, terlihat sebuah tren yang menunjukkan semakin besar ukuran berkas halaman *web* maka semakin baik pula performa dari Progressive Web Apps dibandingkan dengan Mobile Web, dimana pada halaman sedang dan berat Progressive Web Apps justru lebih unggul dari Mobile Web. Berdasarkan tren ini dapat dilihat bahwa Progressive Web Apps akan lebih baik diimplementasikan pada aplikasi web yang memiliki ukuran berkas yang cukup banyak dan ekspektasi pengaksesan yang lebih dari satu kali, seperti aplikasi web e-commerce yang memiliki banyak fungsionalitas dan gambar, portal berita yang biasanya sering dikunjungi dan lain sebagainya. Namun, belum diketahui sampai ukuran berkas berapa tren ini akan terus berlanjut. Adapun grafik yang menunjukkan tren ini dapat dilihat dalam Gambar 5.2.

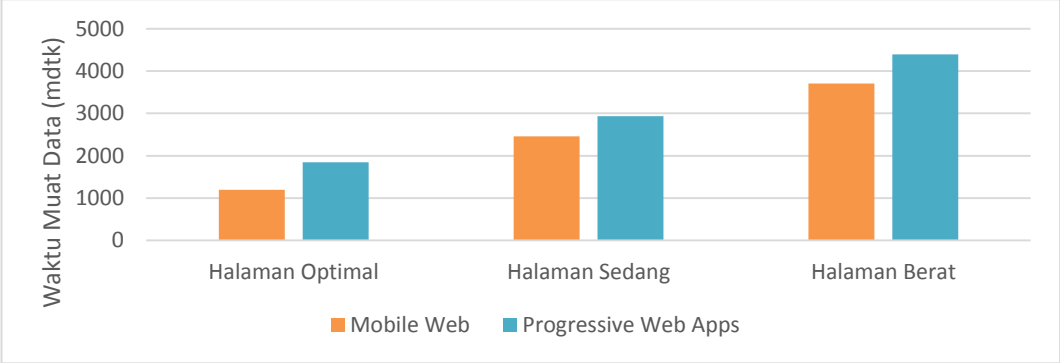


Gambar 5.2 Tren Waktu Tampil Halaman Pengaksesan Ke-2 Skenario 1

Berbeda dengan Waktu Tampil Halaman yang menunjukkan tren positif terhadap Progressive Web Apps, kolom Waktu Muat Data pada pengujian skenario 1 menunjukkan bahwa Progressive Web Apps lebih lambat dalam memuat data-data yang dibutuhkan, dalam hal ini adalah berkas-berkas halaman. Mobile Web unggul pada seluruh aspek baik pada pengaksesan pertama dan lebih baik lagi pada pengaksesan kedua. Berbeda dengan Progressive Web Apps yang justru membutuhkan waktu yang lebih lama pada pengaksesan kedua dibandingkan pada pengaksesan pertama. Salah satu hal yang memungkinkan sebagai penyebab akan hal ini adalah adanya pemanggilan berkas-berkas *cache* yang tersedia di media penyimpanan pada saat halaman dimuat pada pengaksesan kedua. Oleh karena itu, terdapat kemungkinan hal ini diakibatkan oleh kecepatan membaca atau menulis dari media penyimpanan pada perangkat uji atau memang hanya kekurangan yang dimiliki Progressive Web Apps. Adapun grafik yang menunjukkan perbandingan Waktu Muat Data pada pengaksesan ke-1 dapat dilihat dalam Gambar 5.3, sedangkan untuk pengaksesan ke-2 dapat dilihat dalam Gambar 5.4.



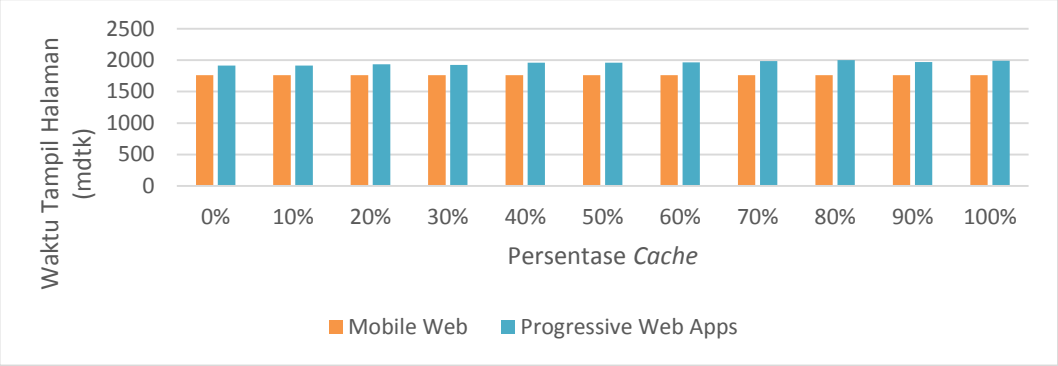
Gambar 5.3 Perbandingan Waktu Muat Data Pengaksesan Ke-1 Skenario 1



Gambar 5.4 Perbandingan Waktu Muat Data Pengaksesan Ke-2 Skenario 1

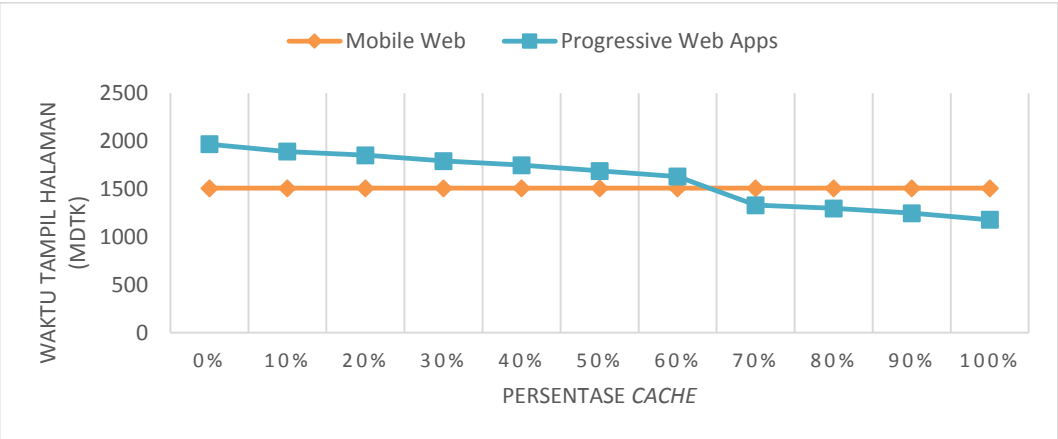
Pada hasil pengujian skenario 2, hasil performa dari waktu respon juga direpresentasikan oleh dua kolom, yaitu Waktu Tampil Halaman dan Waktu Muat Data. Waktu Tampil Halaman pada Progressive Web Apps secara garis besar lebih lambat sekitar 200 mdtk diseluruh ukuran *cache* yang diujikan pada pengaksesan pertama. Sama halnya dengan pengujian skenario 1, ini disebabkan karena Progressive Web Apps membutuhkan proses-proses tambahan yang dilakukan diawal pengaksesan halaman. Namun, yang perlu diperhatikan adalah kisaran

hasil yang sama Waktu Tampil Halaman pada pengaksesan pertama, yaitu dikisaran 1900 mdtk atau 1,9 detik. Hal ini menunjukkan bahwa Waktu Tampil Halaman tidak terpengaruh oleh besaran data yang ingin dijadikan *cache*. Karena mungkin penyimpanan *cache* kedalam media penyimpanan dilakukan setelah pemuatan berkas selesai yang direpresentasikan oleh Waktu Muat Data. Adapun grafik yang menggambarkan hal ini dapat dilihat dalam Gambar 5.5.



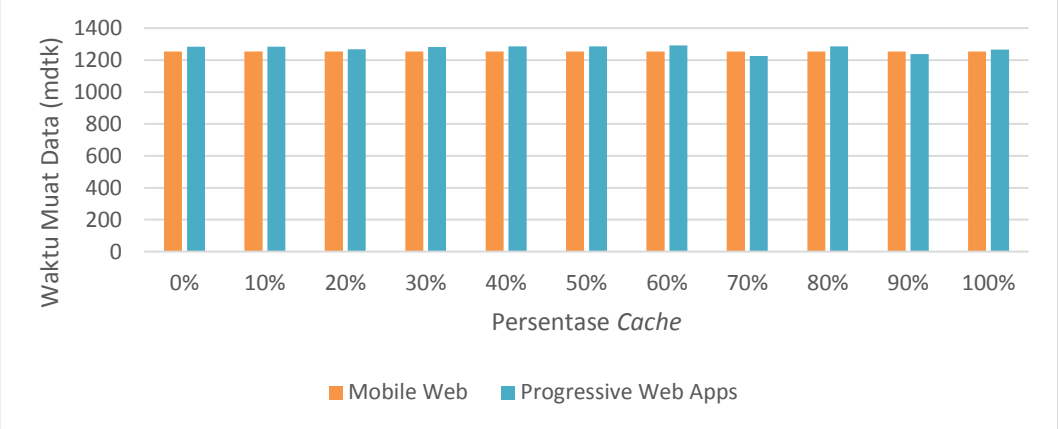
Gambar 5.5 Perbandingan Waktu Tampil Halaman Pengaksesan 1 Skenario 2

Pada pengaksesan kedua oleh Progressive Web Apps, Waktu Tampil Halaman kembali menunjukkan sebuat tren seperti pada skenario 1. Namun tren ini bukan terhadap ukuran berkas halaman tetapi ukuran berkas *cache*, dalam kata lain semakin besar ukuran berkas yang dijadikan *cache* pada suatu halaman semakin baik pula performa pada Waktu Tampil Halaman. Namun, perlu dicatat bahwa apabila dibandingkan dengan Mobile Web, performa Progressive Web Apps pada Waktu Tampil Halaman baru bisa melampaui performa Mobile Web pada persentase *cache* 60% keatas. Selain itu, selisih terbesar yaitu pada persentase *cache* 100% berkisar 300 mdtk dengan Mobile Web. Berdasarkan tren ini dapat dilihat bahwa Progressive Web Apps juga akan lebih baik diimplementasikan pada aplikasi web dengan fungsionalitas yang jauh lebih tinggi dibandingkan dengan kebutuhan akan konten dinamis. Contoh saja pada aplikasi permainan, produktifitas dan lain sebagainya. Adapun grafik yang menunjukkan tren ini dapat dilihat dalam Gambar 5.6.

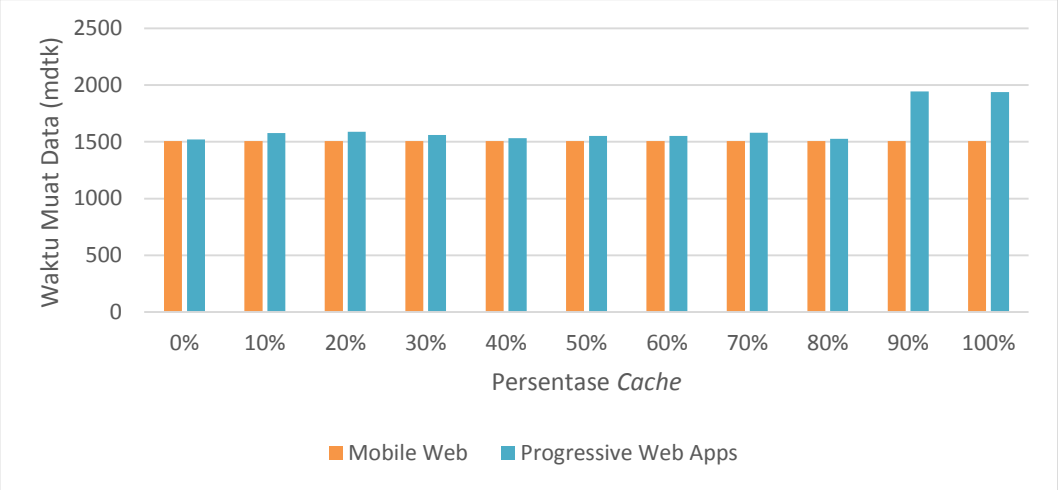


Gambar 5.6 Tren Waktu Tampil Halaman Pengaksesan Ke-2 Skenario 2

Masih pada hasil pengujian skenario 2, Waktu Muat Data berada diangka yang seragam pada hampir semua persentase *cache* yang diujikan, yakni dikisaran 1,2 detik untuk pengaksesan pertama dan 1,5 detik pada pengaksesan kedua. Namun terdapat anomali pada persentase *cache* 90% dan 100% dimana pada pengaksesan kedua berada pada kisaran 1.9 detik. Hal ini mungkin saja disebabkan dimuatnya tipe berkas video dari *cache*, yang mana pada persentase 0% - 80% berkas video tidak dijadikan berkas *cache*. Dibandingkan dengan Mobile Web, Waktu Muat Data untuk pengaksesan pertama pada Progressive Web Apps masih sama pada kisaran 1,2 detik. Sedangkan pada pengaksesan kedua, Progressive Web Apps lebih lambat 300 mdtk yakni dikisaran 1,5 detik. Adapun grafik yang menunjukkan perbandingan Waktu Muat Data skenario 2 pada pengaksesan ke-1 dapat dilihat dalam Gambar 5.7, sedangkan untuk pengaksesan ke-2 dapat dilihat dalam Gambar 5.8.



Gambar 5.7 Perbandingan Waktu Muat Data Pengaksesan Ke-1 Skenario 2



Gambar 5.8 Perbandingan Waktu Muat Data Pengaksesan Ke-2 Skenario 2

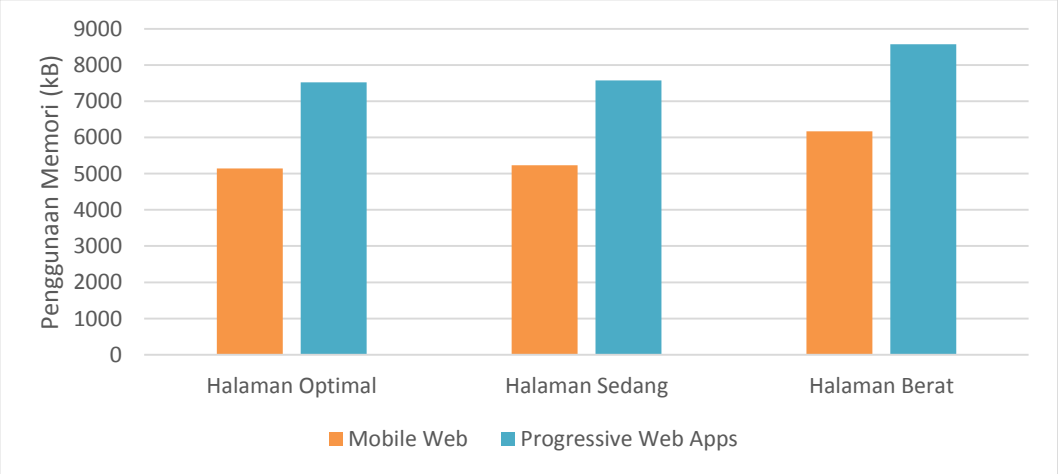
Sama halnya dengan Waktu Tampil Halaman, Waktu Muat Data tidak terpengaruh oleh besaran *cache* pada suatu halaman, tetapi mungkin bisa dipengaruhi oleh tipe berkas yang dijadikan *cache* seperti anomali yang sudah disebutkan. Oleh karena itu sedianya perlu untuk dilakukan penelitian mengenai efisiensi tipe-tipe berkas yang ingin dijadikan *cache*.

5.2.2 Analisis hasil pengujian penggunaan memori

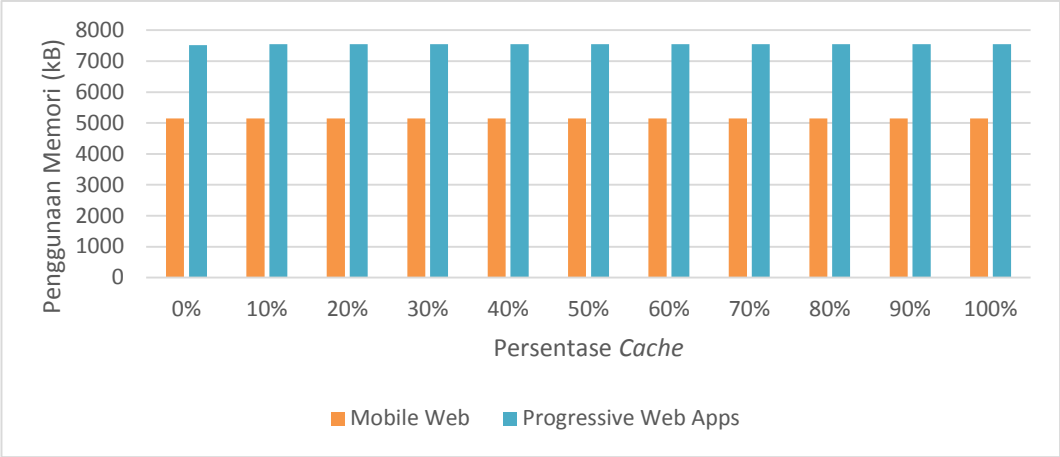
Pada pengujian skenario 1 dan pengujian skenario 2, total penggunaan memori oleh Mobile Web dan Progressive Web Apps menunjukkan pola yang sama. Dimana penggunaan memori pada Progressive Web Apps selalu lebih besar dibandingkan dengan penggunaan memori pada Mobile Web. Hal ini dikarenakan pada Progressive Web Apps terdapat dua proses yang berjalan di memori perangkat, yaitu proses pada halaman *web* dan proses Service Worker. Selain itu, selisih penggunaan memori baik pada tiga fungsional halaman yang diujikan pada skenario 1 maupun seluruh persentase *cache* yang yang diujikan pada skenario 2 juga sama, yakni dikisaran 2300 kB. Dari hasil ini dapat dikatakan bahwa pemilihan tipe aplikasi Progressive Web Apps atau Mobile Web untuk diimplementasikan akan kurang efektif apabila hanya melihat dari performa penggunaan memori, dikarenakan selisih penggunaan pada dua tipe aplikasi yang diuji tidak terlalu signifikan. Adapun grafik yang menunjukkan perbandingan penggunaan memori pada skenario 1 dapat dilihat dalam Gambar 5.9, sedangkan perbandingan penggunaan memori pada skenario 2 dapat dilihat dalam Gambar 5.10

Pada pengujian skenario 1, selisih penggunaan memori pada halaman optimal dan sedang tidak terlampau jauh yakni dikisaran 100 kB baik pada Mobile Web maupun Progressive Web Apps. Tetapi, hal ini tidak terjadi pada selisih antara halaman sedang dan berat dimana selisih berada dikisaran 1000 kB. Tidak diketahui secara pasti alasan dibalik hal ini. Terdapat kemungkinan hal ini disebabkan lagi oleh tipe-tipe berkas yang dimuat, khususnya pada halaman berat. Dikarenakan hal ini terjadi baik pada Mobile Web maupun Progressive Web Apps.

Tidak ada selisih perbedaan penggunaan memori yang signifikan pada pengaksesan pertama dan kedua, dimana rata-rata selisih perbedaan hanya dikisaran kurang dari 50 kB baik pada Mobile Web maupun Progressive Web Apps. Hal ini menunjukkan bahwa penggunaan memori tidak terpengaruh oleh urutan pengaksesan, khususnya pada Progressive Web Apps yang memuat berkas *cache* terlebih dahulu pada pengaksesan kedua dan seterusnya.



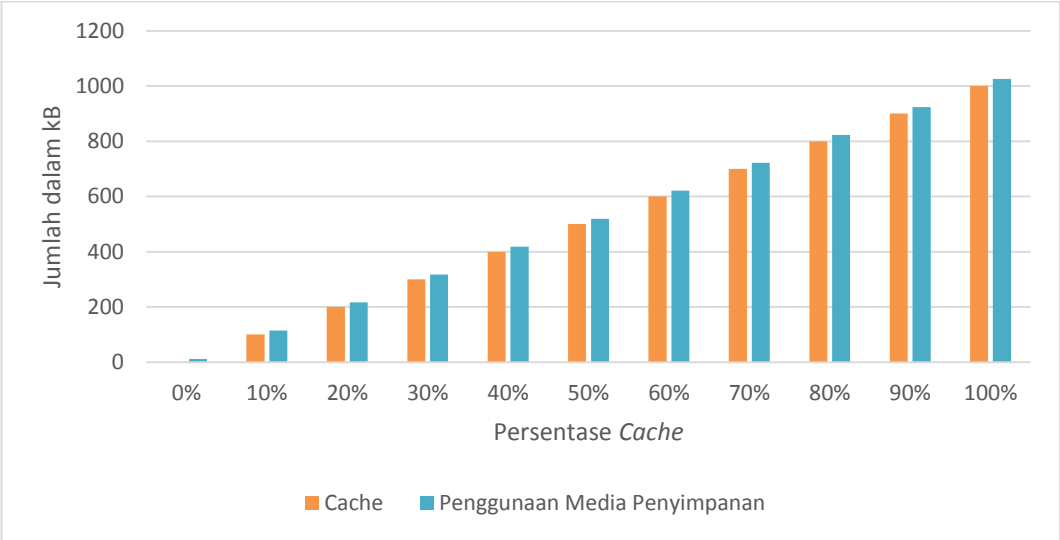
Gambar 5.9 Perbandingan Penggunaan Memori pada Skenario 1



Gambar 5.10 Perbandingan Penggunaan Memori pada Skenario 2

5.2.3 Analisis hasil pengujian penggunaan media penyimpanan

Pada pengujian skenario 1 dan pengujian skenario 2, total penggunaan media penyimpanan oleh Mobile Web tidak berubah, yakni 0 kB dalam arti tidak menggunakan ruang media penyimpanan. Sedangkan pada Progressive Web Apps total penggunaan media penyimpanan menyesuaikan dengan besaran ukuran berkas yang dijadikan *cache* ditambah dengan berkas-berkas lain. Oleh karena itu, pengetahuan yang lebih akan batasan-batasan penggunaan media penyimpanan permban dan kebijaksanaan akan pemilihan berkas yang akan dijadikan *cache* akan sangat membantu dalam menghasilkan baik atau buruknya performa pada Progressive Web Apps. Adapun grafik yang menunjukkan kesesuaian ukuran *cache* dengan penggunaan media penyimpanan dapat dilihat dalam Gambar 5.11. Berkas-berkas lain tersebut adalah service-worker.js dan app.js yang menjadi pondasi dari tipe aplikasi Progressive Web Apps.



Gambar 5.11 Penggunaan Media Penyimpanan pada Skenario 2

Pada hasil pengujian skenario 1, terdapat selisih total penggunaan media penyimpanan pada pengaksesan ke 1 dan ke 2 yang hanya berkisar 1 kB. Tidak

diketahui apa yang menyebabkan perbedaan ini, tetapi perbedaan ini tidaklah terlalu signifikan apabila melihat total penggunaan media penyimpanan yang digunakan. Sebagai catatan, total penggunaan media penyimpanan pada Progressive Web Apps berkisar diangka 67 kB pada halaman optimal, 94 kB pada halaman sedang dan 124 kB pada halaman berat.

Berbeda dengan hasil pengujian skenario 1, pada hasil pengujian skenario 2 terdapat selisih yang lebih besar pada total penggunaan media penyimpanan pada pengaksesan ke 1 dan ke 2. Selisih tersebut sebesar 30 kB yang mana terjadi pada hampir seluruh persentase *cache* yang diujikan, kecuali pada persentase *cache* 0%. Pada persentase *cache* 0% tidak terdapat selisih antara pengaksesan ke 1 dan ke 2, yakni hanya 11 kB yang merupakan berkas service-worker.js dan app.js. Perbedaan pada persentase *cache* 0% dengan persentase lainnya dikarenakan tidak adanya berkas yang dijadikan *cache* pada persentase tersebut.

Selisih antara pengaksesan pertama dan kedua baik pada skenario 1 maupun 2 terlihat ganjil, karena secara logika pada pengaksesan pertama seluruh berkas yang dijadikan *cache* sudah disimpan pada media penyimpanan dan seharusnya pada pengaksesan kedua berkas hanya tinggal dipanggil kembali dari media penyimpanan, tidak perlu menyimpan *cache* yang lain; dalam arti total *cache* pada media penyimpanan seharusnya tetap sama. Tetapi yang perlu dicatat bahwa penggunaan media penyimpanan pada Progressive Web Apps hampir berbanding lurus dengan berkas-berkas yang dijadikan *cache* pada suatu halaman.

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan penelitian yang dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Performa terkait waktu respon menyesuaikan dengan ukuran berkas dan *cache* yang digunakan serta frekuensi pengaksesan halaman aplikasi. Pada ukuran berkas dan *cache* yang kecil Mobile Web masih lebih unggul dibandingkan dengan Progressive Web Apps, sedangkan pada ukuran berkas dan *cache* yang cukup besar Progressive Web Apps mampu mengungguli Mobile Web. Perlu dicatat bahwa performa dari Progressive Web Apps juga akan lebih baik apabila frekuensi pengaksesan yang tidak hanya sesekali saja.
2. Performa terkait penggunaan memori pada Mobile Web lebih kecil dibandingkan dengan Progressive Web Apps. Selisih penggunaan memori antara Mobile Web dan Progressive Web Apps pada penelitian ini tidak terlampau jauh, yakni sekitar 2300 kB. Dimana selisih tersebut disebabkan oleh adanya proses tambahan pada Progressive Web Apps, yaitu Service Worker.
3. Performa terkait penggunaan media penyimpanan pada Mobile Web lebih unggul dibandingkan dengan Progressive Web App, dimana pada Mobile Web Tidak Menggunakan Ruang Penyimpanan Sama Sekali. Namun, Keunggulan Dari Mobile Web tidak terlampau jauh dari Progressive Web Apps menyesuaikan dengan ukuran berkas *cache* yang disimpan. Pada Progressive Web Apps penggunaan media penyimpanan terdiri dari berkas *cache* yang disimpan ditambah beberapa berkas fungsional Progressive Web Apps yang ukurannya terbilang kecil.

6.2 Saran

Pada penelitian ini tentu masih banyak kekurangan-kekurangan yang belum dapat dipenuhi dengan baik. Sehingga perlu adanya penelitian lebih lanjut agar penelitian ini dapat lebih bermanfaat kedepannya. Adapun saran-saran yang dapat dilakukan pada penelitian selanjutnya adalah sebagai berikut:

1. Penambahan skenario terkait tipe-tipe berkas yang diujikan untuk dapat mengetahui penyebab munculnya anomali-anomali yang muncul dibeberapa parameter pengujian pada skenario yang diujikan.
2. Penambahan beberapa halaman dengan ukuran berkas yang lebih besar dan atau beberapa halaman dengan selisih ukuran berkas yang lebih kecil lagi sehingga mendapatkan hasil yang lebih spesifik dan lebih akurat dikarenakan belum terlihatnya perbedaan performa yang cukup signifikan, serta agar dapat mengetahui keberlangsungan tren-tren yang muncul pada hasil penelitian.

DAFTAR PUSTAKA

Akamai Technologies, 2009. *Akamai Reveals 2 Seconds As The New Threshold Of Acceptability For ECommerce Web Page Response Times*. [online] Tersedia di: <https://www.akamai.com/us/en/about/news/press/2009-press/akamai-reveals-2-seconds-as-the-new-threshold-of-acceptability-for-ecommerce-web-page-response-times.jsp> [Diakses 28 Januari 2017]

AppDynamics, 2014. *Mobile App Performance Explained*. [online] Tersedia di: <https://kapost-files-prod.s3.amazonaws.com/published/54dd1e53715cba88750000e4/white-paper-mobile-app-performance-explained.pdf> [Diakses 3 Maret 2017]

Archibald, J., 2014. *The offline cookbook*. [online] Tersedia di: <https://jakearchibald.com/2014/offline-cookbook> [Diakses 14 Maret 2017]

Archibald, J. [Google Developers], 10 Juni 2015. *Supercharging page load (100 Days of Google Dev) .* [video] Tersedia di: <https://www.youtube.com/watch?v=d5_6yHixpsQ> [Diakses 24 Desember 2016]

Basques, K., 2017. *How to Use the Timeline Tool*. [online] Tersedia di: <https://developers.google.com/web/tools/chrome-devtools/evaluate-performance/timeline-tool> [Diakses 23 Desember 2017]

Basques, K., & Kearney, M., 2017. Measure Resource Loading Times. [online] Tersedia di: <https://developers.google.com/web/tools/chrome-devtools/network-performance/resource-loading> [Diakses 23 Desember 2017]

Belson, D., 2016. *Akamai's [state of the internet] – Q2 2016 Report, 9(2), p.43*.

BlueFountainMedia, 2015. [online] *Mobile Website vs. Native App vs. Mobile Web App*. Tersedia di: <https://www.bluefountainmedia.com/blog/mobile-app/> [Diakses 3 September 2016]

Chrome Developer, n.d. *Chrome DevTools Overview*. [online] Tersedia di: <https://developer.chrome.com/devtools> [Diakses 23 Desember 2017]

Cremin, R., 2016. *The web is Doom*. Tersedia di: <https://mobiforge.com/research-analysis/the-web-is-doom> [Diakses 6 April 2017]

Everts, T., 2015. *Page bloat update: The average web page is more than 2 MB in size*. [online] Tersedia di: <https://www.soasta.com/blog/page-bloat-average-web-page-2-mb/> [Diakses 22 Maret 2017]

Fui, F., & Nah, H., 2012. *A study on tolerable waiting time: How long are web users willing to wait?*. Lincoln: University of Nebraska-Lincoln.

Gaunt, M., & Kinlan, P., 2017. *The Web App Manifest*. [online] Tersedia di <https://developers.google.com/web/fundamentals/web-app-manifest/> [Diakses 23 Desember 2017]

Google Developers, 2017. Progressive Web Apps. [online] Tersedia di: <https://developers.google.com/web/progressive-web-apps/> [Diakses 1 Maret 2017].

International Telecommunication Union (ITU), 2015. *ICT Facts & Figures - The world in 2015*. Geneva: ITU.

Krauss, A., 2016. [online] How Browser Caching Works. Tersedia di: < https://thesocietea.org/2016/05/how-browser-caching-works/> [Diakses 15 Januari 2018]

LePage, P., 2016. [online] *Your First Progressive Web App*. Tersedia di: <https://developers.google.com/web/fundamentals /getting-started/your-first-progressive-web-app/> [Diakses 2 September 2016]

Liu, C., 2015. *Worldwide Internet And Mobile Users: eMarketer's Updated Estimates for 2015*. New York: eMarketer Inc.

Lynch, M., 2016. *What are Progressive Web Apps?*. [online] Tersedia di: < http://blog.ionic.io/what-is-a-progressive-web-app/> [Diakses 2 September 2016]

Miller R, B., 1968. *Response Time in Man-Computer Conversational Transaction (Periodical style)*, Ipsj Magazine, vol. 11, pp. 53 – 54

Mozilla, 2016. *JavaScript.* [online] Tersedia di <https://developer.mozilla.org/en-US/docs/Web/JavaScript > [Diakses 20 September 2016]

Mozilla, 2017. *Promise*. [online] Tersedia di: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise> [Diakses 23 Desember 2017]

Nurhaman, 2016. [online] *HTML 1 – Pengenalan*. Tersedia di <http://wilayahilmu.com/read/html-1---pengenalan> [Diakses 3 September 2016]

Osmani, A., 2016. *Offline Storage for* Progressive Web Apps. [online] Tersedia di: <https://medium.com/dev-channel/offline-storage-for-progressive-web-apps-70d52695513c> [Diakses 14 Maret 2017]

Osmani, A., 2017. *The App Shell Model*. [online] Tersedia di <https://developers.google.com/web/fundamentals/architecture/app-shell> [Diakses 23 Desember 2017]

Roy-Chowdury, R. [Google Developers], 18 Mei 2016. *Google I/O 2016 - Keynote* . [video] Tersedia di: <https://www.youtube.com/watch?v=862r3XS2YB0>

Russel, A., 2015. Progressive Web Apps*: Escaping Tabs Without Losing Our Soul*. [online] Tersedia di: <https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/> [Diakses 1 Februari 2017]

Semenov, A., 2017. *How Progressive Web Apps make the Web great again*. [online] Tersedia di: <https://webagility.com/posts/how-progressive-web-apps-make-the-web-great-again> [Diakses 1 November 2017]

SmartBear, 2017. [online] *Web Page Load Time*. Tersedia di: <https://support.smartbear.com/alertsite/docs/monitors/metrics/web-page-load-time.html> [Diakses 18 Desember 2017]

W3, 1996. *What is CSS?*. [online] Tersedia di <*https://www.w3.org/Style/CSS/Overview.en.html*> [Diakses 16 September 2016].